

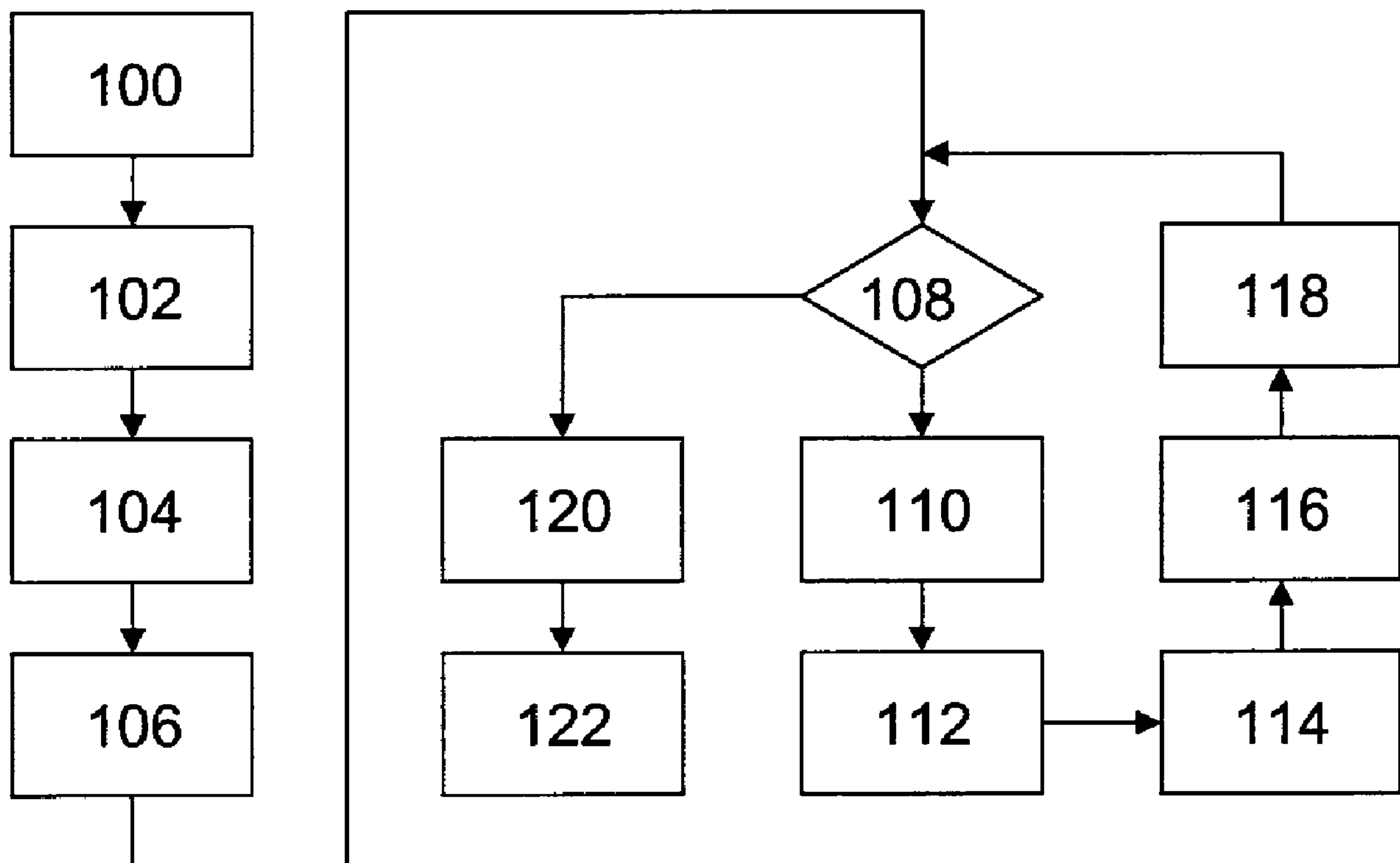


(86) Date de dépôt PCT/PCT Filing Date: 2009/01/23
(87) Date publication PCT/PCT Publication Date: 2009/09/17
(85) Entrée phase nationale/National Entry: 2010/07/14
(86) N° demande PCT/PCT Application No.: FR 2009/000071
(87) N° publication PCT/PCT Publication No.: 2009/112686
(30) Priorité/Priority: 2008/01/23 (FR08/00344)

(51) Cl.Int./Int.Cl. *G06F 7/72* (2006.01)
(71) Demandeur/Applicant:
INSIDE CONTACTLESS, FR
(72) Inventeurs/Inventors:
BENTEO, BRUNO, FR;
FEIX, BENOIT, FR;
NEROT, SEBASTIEN, FR
(74) Agent: GOUDREAU GAGE DUBUC

(54) Titre : PROCEDE ET DISPOSITIFS DE CONTRE-MESURE POUR CRYPTOGRAPHIE ASYMETRIQUE
(54) Title: COUNTERMEASURE METHOD AND DEVICES FOR ASYMMETRIC CRYPTOGRAPHY

Figure 4



(57) **Abrégé/Abstract:**

Ce procédé de contre-mesure dans un composant électronique mettant en oeuvre un algorithme de cryptographie asymétrique à clé privée, comprend la génération (100) d'un paramètre de protection et le calcul (104), à l'aide d'une primitive, d'une donnée

(57) Abrégé(suite)/Abstract(continued):

intermédiaire à partir du paramètre de protection. Il comprend en outre les étapes consistant à diviser (110) la représentation binaire de la clé privée en plusieurs blocs binaires, à transformer (112) chaque bloc binaire à l'aide du paramètre de protection et, pour chaque bloc binaire transformé, à effectuer (114) un calcul intermédiaire à l'aide de la primitive, et à calculer (106-122) une donnée de sortie par combinaison (116) de la donnée intermédiaire avec les calculs intermédiaires (114).

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

(19) Organisation Mondiale de la Propriété
Intellectuelle
Bureau international(10) Numéro de publication internationale
WO 2009/112686 A3(43) Date de la publication internationale
17 septembre 2009 (17.09.2009)(51) Classification internationale des brevets :
G06F 7/72 (2006.01)(21) Numéro de la demande internationale :
PCT/FR2009/000071(22) Date de dépôt international :
23 janvier 2009 (23.01.2009)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
08/00344 23 janvier 2008 (23.01.2008) FR(71) Déposant (pour tous les États désignés sauf US) :
INSIDE CONTACTLESS [FR/FR]; 41 Parc Club du
Golf, F-13856 Aix En Provence Cedex 3 (FR).

(72) Inventeurs; et

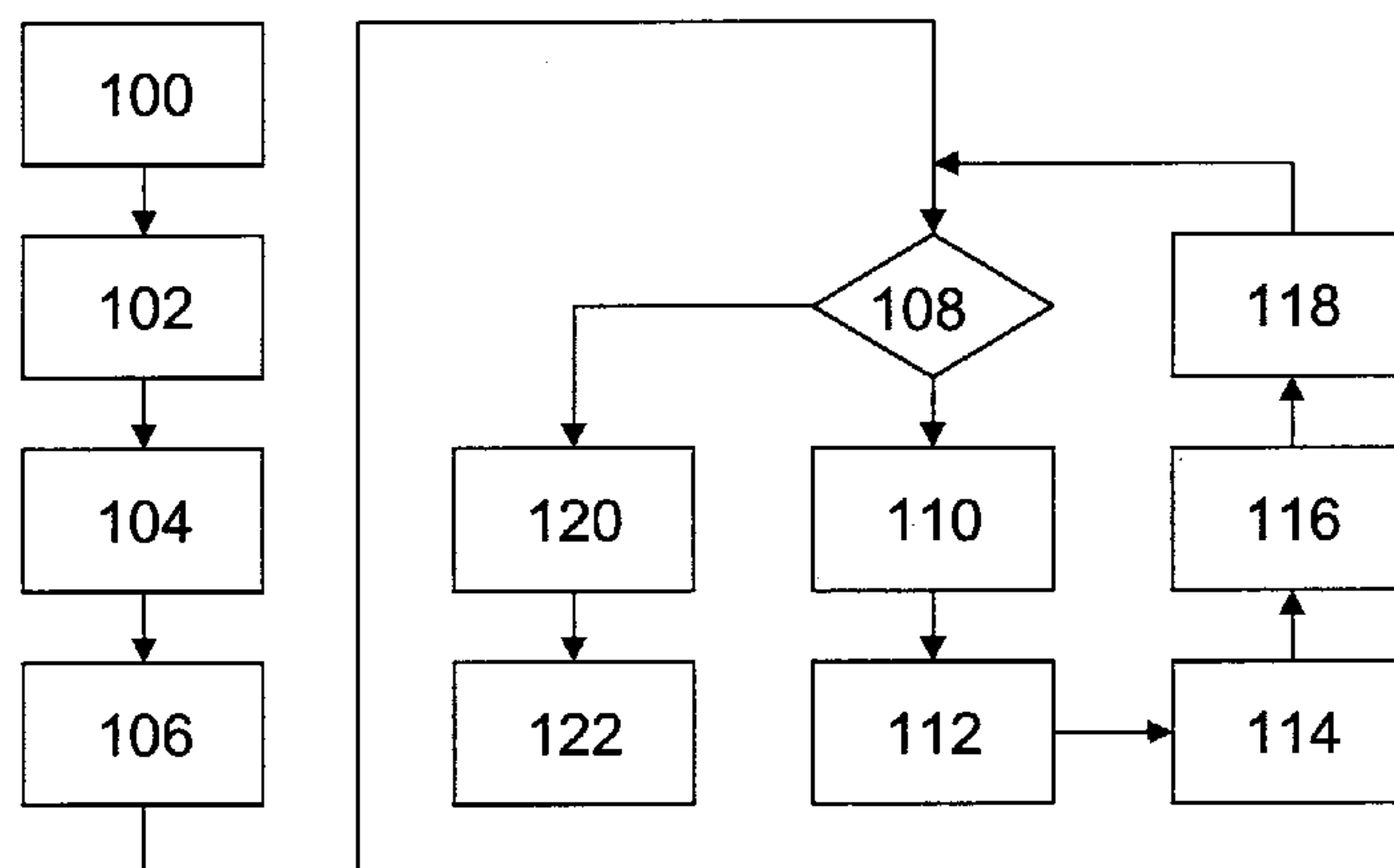
(75) Inventeurs/Déposants (pour US seulement) : **BENTEO,**
Bruno [FR/FR]; 11, rue Bernard Mule, F-31400
Toulouse (FR). **FEIX, Benoît** [FR/FR]; Résidence
Cytharista., Bât. B, Avenue Cytharista, F-13600 La Ciotat
(FR). **NEROT, Sébastien** [FR/FR]; 420 Nouveau
Chemin du Cannet, F-13490 Jouques (FR).(74) Mandataires : **MARCHAND, André** et al.; OMNIPAT,
24 Place des Martyrs de la Résistance, F-13100 Aix en
Provence (FR).(81) États désignés (sauf indication contraire, pour tout titre
de protection nationale disponible) : AE, AG, AL, AM,
AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ,
CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ,
EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN,
HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR,
KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME,
MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO,
NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG,
SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA,
UG, US, UZ, VC, VN, ZA, ZM, ZW.(84) États désignés (sauf indication contraire, pour tout titre
de protection régionale disponible) : ARIPO (BW, GH,
GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,
ZW), eurasienn (AM, AZ, BY, KG, KZ, MD, RU, TJ,
TM), européen (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV,
MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR),
OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML,
MR, NE, SN, TD, TG).

Déclarations en vertu de la règle 4.17 :

[Suite sur la page suivante]

(54) Title : COUNTERMEASURE METHOD AND DEVICES FOR ASYMMETRIC CRYPTOGRAPHY

(54) Titre : PROCÉDE ET DISPOSITIFS DE CONTRE-MESURE POUR CRYPTOGRAPHIE ASYMETRIQUE

Figure 4

(57) Abstract : The invention relates to a countermeasure method in an electronic component that uses a private-key asymmetric cryptography algorithm, and comprises generating (100) a protection parameter and calculating (104), using a primitive, an intermediate data from said protection parameter. The method further comprises the steps of splitting (110) the binary representation of the private key into a plurality of binary units, converting (112) each binary unit using the protection parameter and, for each converted binary unit, carrying out (114) an intermediate calculation using the primitive, and calculating (106-122) an output datum by combining (116) the intermediate data with the intermediate calculations (114).

(57) Abrégé :

[Suite sur la page suivante]

WO 2009/112686 A3

WO 2009/112686 A3

-
- *relative au droit du déposant de revendiquer la priorité de la demande antérieure (règle 4.17.iii)* — *avant l'expiration du délai prévu pour la modification des revendications, sera republiée si des modifications sont reçues (règle 48.2.h)*
- *relative à la qualité d'inventeur (règle 4.17.iv)*

Publiée :

- *avec rapport de recherche internationale (Art. 21(3))*

(88) Date de publication du rapport de recherche internationale :

14 janvier 2010

Ce procédé de contre-mesure dans un composant électronique mettant en œuvre un algorithme de cryptographie asymétrique à clé privée, comprend la génération (100) d'un paramètre de protection et le calcul (104), à l'aide d'une primitive, d'une donnée intermédiaire à partir du paramètre de protection. Il comprend en outre les étapes consistant à diviser (110) la représentation binaire de la clé privée en plusieurs blocs binaires, à transformer (112) chaque bloc binaire à l'aide du paramètre de protection et, pour chaque bloc binaire transformé, à effectuer (114) un calcul intermédiaire à l'aide de la primitive, et à calculer (106-122) une donnée de sortie par combinaison (116) de la donnée intermédiaire avec les calculs intermédiaires (114).

PROCEDE ET DISPOSITIFS DE CONTRE-MESURE POUR CRYPTOGRAPHIE ASYMETRIQUE

La présente invention concerne un procédé de contre-mesure dans un composant électronique mettant en œuvre un algorithme de cryptographie asymétrique à clé privée, résistant à des attaques visant à découvrir la clé privée. Elle concerne également un dispositif à microcircuit et
5 un dispositif portable, notamment une carte à puce, mettant en œuvre un tel procédé.

Comme cela est illustré sur la figure 1, une application algorithmique de cryptographie asymétrique 10 impliquant l'utilisation d'une clé privée d est généralement mise en œuvre par un microcircuit 12 pour authentifier
10 l'émission d'un message M par une signature de ce message ou pour protéger la réception d'un message M chiffré par un déchiffrement de ce message, à l'aide de la clé privée. La clé privée d est par exemple stockée dans le microcircuit 12 qui comporte une mémoire 14 incluant elle-même un espace de mémoire sécurisé 16 prévu à cet effet et un microprocesseur 18
15 pour exécuter l'algorithme de cryptographie asymétrique 10.

Les dispositifs à microcircuits mettant en œuvre des algorithmes de cryptographie font parfois l'objet d'attaques visant à déterminer les données secrètes qu'ils manipulent telles que la ou les clés utilisées et éventuellement, dans certains cas, de l'information sur les messages eux-mêmes. En particulier, les algorithmes de cryptographie asymétrique font
20 l'objet d'attaques visant à découvrir la clé privée, lorsque celle-ci est utilisée. Les attaques par canaux auxiliaires constituent une famille importante des techniques de cryptanalyse qui exploitent certaines propriétés des implémentations logicielles ou matérielles des algorithmes de cryptographie.

Parmi les attaques par canaux auxiliaires connues, les attaques de type SPA (de l'anglais « Simple Power Analysis ») ou DPA (de l'anglais « Differential Power Analysis ») consistent à mesurer les courants et tensions entrants et sortants dans le microcircuit au cours de l'exécution de l'algorithme de cryptographie asymétrique dans le but d'en déduire la clé
25 privée. La faisabilité de cette famille d'attaques a été démontrée dans l'article de P. Kocher, J. Jaffe et B. Jun intitulé « Differential Power Analysis » notamment publié dans Advances in Cryptology - Crypto 99 Proceedings, Lecture Notes In Computer Science Vol. 1666, M. Wiener, ed., Springer-Verlag, 1999.
30

Les attaques temporelles analysent le temps mis pour effectuer certaines opérations. De telles attaques sur des algorithmes de cryptographie asymétrique sont décrites dans l'article de P. Kocher, N. Koblitz intitulé « Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems » notamment publié dans Advances in Cryptology - Crypto 96, 16th annual international cryptology conference, Aug. 18-22, 1996 Proceedings.

On connaît également les attaques par injection de faute(s), parmi lesquelles on peut citer les attaques DFA (de l'anglais « Differential Fault Analysis »), qui consistent à engendrer volontairement des fautes lors de l'exécution de l'algorithme de cryptographie, par exemple en perturbant le microcircuit sur lequel il s'exécute. Une telle perturbation peut inclure un (ou plusieurs) éclairage(s) bref(s) du microcircuit ou la génération d'un ou plusieurs pic(s) de tension sur l'un de ses contacts. Elle permet ainsi sous certaines conditions d'exploiter les erreurs de calcul et de comportement générées afin d'obtenir une partie voire la totalité de la clé privée recherchée.

En particulier, lors de l'exécution de l'algorithme de cryptographie asymétrique connu sous le nom de RSA (du nom de ses auteurs Rivest, Shamir et Adleman), une primitive consistant en une exponentiation modulaire est exécutée. Une implémentation efficace de cette primitive utilise une représentation binaire de la clé privée d en réalisant des itérations sur chaque bit de cette représentation binaire. Dans chaque itération, le calcul effectué et de fait la consommation d'énergie lors de la réalisation du calcul dépendent de la valeur du bit concerné. Par conséquent, l'exécution d'une telle primitive rend la clé privée particulièrement vulnérable aux attaques précitées. De même, lors de l'exécution d'une adaptation de cet algorithme de cryptographie asymétrique utilisant une courbe elliptique, une primitive consistant en une multiplication scalaire est exécutée. Une implémentation efficace de cette primitive utilise une représentation binaire de la clé privée d en réalisant des itérations sur chaque bit de cette représentation binaire. De même, dans chaque itération, la consommation d'énergie lors de la réalisation du calcul dépend de la valeur du bit concerné. Par conséquent, l'exécution d'une telle primitive rend aussi la valeur du scalaire, qui peut être assimilé pour des raisons de sécurité à une clé privée, particulièrement vulnérable aux attaques.

Afin de lutter contre ces attaques qui sont variées par nature, de nombreuses solutions très différentes les unes des autres ont été apportées. L'invention concerne plus particulièrement celles qui mettent en œuvre un procédé de contre-mesure dans un composant électronique mettant en œuvre un algorithme de cryptographie asymétrique à clé privée, comprenant les étapes consistant à :

- générer un paramètre de protection,
- calculer, à l'aide d'une primitive de l'algorithme de cryptographie, une donnée intermédiaire à partir d'une donnée d'entrée et du paramètre de protection.

Ces algorithmes prévoient en général de transformer la clé privée à l'aide du paramètre de protection généré, d'appliquer la primitive à la clé privée transformée et de combiner le résultat obtenu avec la donnée intermédiaire.

Généralement, le paramètre de protection a est généré de façon classique à l'aide d'un générateur de données pseudo aléatoires, de sorte que l'exécution de la primitive par l'algorithme de cryptographie est elle-même rendue aléatoire et décorrélée de la clé privée utilisée, par exemple par une technique communément qualifiée de masquage, qui peut être également renommée méthode de transformation ou de déformation des données puisque leur manipulation est déformée par opposition à leur utilisation brute, réalisée, par une section de contre-mesure du microprocesseur, à l'aide du paramètre de protection a . Ainsi, les données intermédiaires de l'algorithme de cryptographie et, par suite, les courants mesurables sont modifiés par le paramètre de protection aléatoire et leur observation ne permet pas de retrouver la véritable valeur de la clé privée. En revanche, le masquage ne perturbe pas l'algorithme lui-même qui fournit donc le même résultat avec ou sans masquage.

Un procédé de ce type est par exemple décrit dans le brevet Américain délivré sous le numéro US 6,381,699.

Dans ce document, un mode de réalisation dans le domaine de la cryptographie asymétrique de type RSA est décrit en référence à la figure 3. Dans l'algorithme RSA à clé publique e et à clé privée d , pour effectuer une signature ou un déchiffrement, exécuter la primitive consiste à calculer une donnée de sortie S à partir d'une donnée d'entrée M et de la clé privée d de la façon suivante :

$S = M^d \bmod N$, où N est le module RSA, produit de deux entiers premiers secrets, et où e et d vérifient la relation $e.d = \phi(N)$, la fonction $\phi(.)$ représentant la fonction indicatrice d'Euler.

En notant $[d_{n-1}, \dots, d_0]_2$ la représentation binaire de la clé privée d , ce calcul peut être effectué comme suit :

$$S = 1$$

Pour i variant de $n-1$ à 0 :

$$S \leftarrow S^2 \bmod N$$

$$\text{si } d_i = 1, S \leftarrow S \times M \bmod N$$

Le mode de réalisation d'un algorithme RSA résistant aux attaques décrit dans le document US 6,381,699, comporte une première étape 300 lors de laquelle un paramètre de protection $d1$ est généré de la façon suivante : on génère un nombre premier k aléatoirement choisi tel que $0 < k < 2^{128}$, puis $z = k.\phi(n)$, puis on choisit aléatoirement $d1$ tel que $0 < d1 < z$ et $\text{pgcd}(d1, z) = 1$ (pgcd est la fonction « plus grand dénominateur commun »).

La clé privée est alors transformée de la façon suivante : $d2 = d \times (d1^{-1} \bmod z) \bmod z$.

Suite à la réception de la donnée d'entrée M , de nouvelles transformations sont réalisées sur $d1$ et $d2$ avant de procéder aux deux calculs suivants (étapes 345 et 350) :

- $S_0 = M^{d1} \bmod N$ (calcul à l'aide de la primitive d'une donnée intermédiaire S_0 à partir de la donnée d'entrée M et du paramètre de protection $d1$),

- $S = S_0^{d2} \bmod N$ (calcul de la donnée de sortie par combinaison de la donnée intermédiaire S_0 avec l'application de la primitive à la clé privée transformée $d2$).

Un autre mode de réalisation d'un algorithme RSA résistant aux attaques, plus simple mais également décrit dans le document US 6,381,699, comporte une première étape lors de laquelle le paramètre de protection $d1$ est choisi aléatoirement tel que $0 < d1 < d$.

La clé privée est alors transformée de la façon suivante : $d2 = d - d1$.

Suite à la réception de la donnée d'entrée M , de nouvelles transformations sont réalisées sur $d1$ et $d2$ avant de procéder aux deux calculs suivants :

- $S_1 = M^{d1} \bmod N$ (calcul à l'aide de la primitive d'une donnée intermédiaire S_1 à partir de la donnée d'entrée M et du paramètre de protection $d1$),

- $S_2 = M^{d_2} \bmod N$, $S = S_1 \cdot S_2 \bmod N$ (calcul de la donnée de sortie S par combinaison de la donnée intermédiaire S_1 avec l'application S_2 de la primitive à la clé privée transformée d_2).

Dans chacun des deux cas précités de l'état de la technique, on
5 décompose la clé privée d en au moins deux exposants d_1 et d_2 de tailles comparables à celle de d , de sorte que l'on complique l'algorithme RSA en imposant au moins deux exécutions de l'exponentiation modulaire au lieu d'une. Bien que l'on ait ainsi conçu un algorithme de cryptographie asymétrique résistant à certaines attaques par canaux auxiliaires, c'est au
10 prix d'une complexité sensiblement accrue de l'implémentation puisque que la complexité est en fait doublée.

Il peut ainsi être souhaité de prévoir un procédé de cryptographie asymétrique résistant aux attaques du type précité et qui soit simple à mettre en œuvre.

15 Un mode de réalisation de l'invention concerne un procédé de contre-mesure dans un composant électronique mettant en œuvre un algorithme de cryptographie asymétrique à clé privée, comprenant les étapes consistant à :

- générer un paramètre de protection,
- calculer, à l'aide d'une primitive de l'algorithme de cryptographie,
20 une donnée intermédiaire à partir d'une donnée d'entrée et du paramètre de protection,

caractérisé en ce qu'il comporte en outre les étapes consistant à :

- diviser la représentation binaire de la clé privée en plusieurs blocs binaires,
- 25 - transformer chaque bloc binaire à l'aide du paramètre de protection et, pour chaque bloc binaire transformé, effectuer un calcul intermédiaire à l'aide de la primitive, et
- calculer une donnée de sortie par combinaison de la donnée intermédiaire avec les calculs intermédiaires.

30 Ainsi le paramètre de protection est utilisé pour transformer les blocs binaires plutôt que la représentation binaire complète de la clé privée. Par conséquent, la taille de la représentation binaire du paramètre de protection peut être nettement inférieure à celle de la représentation binaire de la clé privée, c'est-à-dire de l'ordre de celle des blocs binaires. On simplifie
35 d'autant le calcul car, même si l'on augmente le nombre d'exécutions de la primitive, celles-ci opèrent sur des données binaires de plus petites tailles. Au bout du compte, on peut de la sorte protéger l'exécution de l'algorithme

de cryptographie asymétrique en diminuant sa complexité de façon sensible par rapport aux procédés de contre-mesure classiques.

Selon un mode de réalisation, le procédé de contre-mesure comprend l'étape consistant à diviser la représentation binaire de la clé privée de telle sorte que la taille de chaque bloc binaire soit supérieure ou égale à celle de la représentation binaire du paramètre de protection.

Selon un mode de réalisation, le procédé de contre-mesure comprend l'étape consistant à diviser la représentation binaire de la clé privée en plusieurs blocs binaires de telle sorte que la somme des tailles des blocs binaires soit égale à la taille de la représentation binaire de la clé privée.

Selon un mode de réalisation, le procédé de contre-mesure comprend l'étape consistant à déterminer aléatoirement de manière itérative la taille de chaque bloc binaire de telle sorte que la valeur de chaque bloc binaire soit supérieure à la valeur du paramètre de protection.

Selon un mode de réalisation, le procédé de contre-mesure comprend les étapes consistant à :

- choisir la taille k de la représentation binaire du paramètre de protection de telle sorte qu'il existe un entier $u \geq 2$ tel que $n = k.u$, n étant la taille de la représentation binaire de la clé privée, et
- diviser la représentation binaire de la clé privée en u blocs binaires de k bits chacun.

Selon un mode de réalisation, la primitive est une exponentiation modulaire de la donnée d'entrée par la clé privée pour la réalisation d'un algorithme de cryptographie de type RSA ou RSA CRT.

Selon un mode de réalisation, le procédé de contre-mesure comprend une étape préalable consistant à masquer le module RSA et la donnée d'entrée.

Selon un mode de réalisation, la primitive est une multiplication scalaire de la donnée d'entrée par la clé privée, pour la réalisation d'un algorithme de cryptographie basé sur une courbe elliptique dans lequel la donnée d'entrée est un point prédéterminé de la courbe elliptique.

Selon un mode de réalisation, le procédé de contre-mesure comprend une étape préalable consistant à masquer le point prédéterminé de la courbe elliptique.

Selon un mode de réalisation, le procédé de contre-mesure comprend en outre les étapes consistant à :

- générer initialement, de façon reproductible, au moins un paramètre de vérification avant toute exécution de la primitive,

- régénérer ce paramètre de vérification en cours d'exécution ou après exécution de la primitive et comparer le paramètre de vérification régénéré au paramètre de vérification généré initialement.

Selon un mode de réalisation, l'étape de régénération et de comparaison est exécutée à chaque itération de la primitive lorsqu'elle est appliquée à un bloc binaire transformé.

Selon un mode de réalisation, le procédé de contre-mesure comprend l'étape consistant à déclencher une alerte et brouiller au moins la clé privée, si l'étape de régénération et de comparaison indique une différence entre le paramètre de vérification généré initialement et le paramètre de vérification régénéré.

Selon un mode de réalisation, la génération du paramètre de protection et/ou du paramètre de vérification comporte les étapes consistant à :

- définir une fonction génératrice, par applications successives à au moins un paramètre secret prédéterminé et stocké en mémoire, d'une séquence de valeurs déterminable uniquement à partir de ce paramètre secret et de cette fonction,

- générer le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette séquence.

Selon un mode de réalisation, le procédé de contre-mesure comprend les étapes consistant à :

- définir une pluralité de fonctions, chaque fonction étant génératrice, par applications successives à au moins un paramètre secret correspondant prédéterminé et stocké en mémoire, d'une séquence de valeurs correspondante déterminable uniquement à partir du paramètre secret correspondant et de la fonction correspondante,

- combiner la pluralité de séquences de valeurs générées à l'aide d'une relation prédéfinie pour générer une nouvelle séquence de valeurs,

- générer le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence.

Selon un mode de réalisation, le procédé de contre-mesure comprend les étapes consistant à :

- définir une fonction génératrice, par applications successives à au moins un paramètre secret prédéterminé et stocké en mémoire, d'une séquence de valeurs déterminable uniquement à partir du paramètre secret et de la fonction,

5 - combiner la séquence de valeurs générées avec des paramètres publics de l'algorithme de cryptographie pour générer une nouvelle séquence de valeurs,

10 - générer le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence.

Un autre mode de réalisation de l'invention consiste à fournir un dispositif à microcircuit, comprenant un microprocesseur pour la mise en œuvre d'un procédé de contre-mesure d'un algorithme de cryptographie asymétrique à clé privée, au moins une mémoire sécurisée pour le stockage de la clé privée, et un générateur de données pour la génération d'un paramètre de protection, caractérisé en ce qu'il est configuré pour :

15 - calculer, à l'aide d'une primitive de l'algorithme de cryptographie, une donnée intermédiaire à partir d'une donnée d'entrée et du paramètre de protection,

20 - diviser la représentation binaire de la clé privée en plusieurs blocs binaires,

- transformer chaque bloc binaire à l'aide du paramètre de protection et, pour chaque bloc binaire transformé, effectuer un calcul intermédiaire à l'aide de la primitive,

25 - calculer une donnée de sortie par combinaison de la donnée intermédiaire avec les calculs intermédiaires.

Selon un mode de réalisation, le microprocesseur est configuré pour déterminer aléatoirement de manière itérative la taille de chaque bloc binaire de telle sorte que la valeur de chaque bloc binaire soit supérieure à la valeur du paramètre de protection.

30 Selon un mode de réalisation, le générateur de données est configuré pour choisir la taille k de la représentation binaire du paramètre de protection de telle sorte qu'il existe un entier $u \geq 2$ tel que $n = k.u$, n étant la taille de la représentation binaire de la clé privée, et le microprocesseur est configuré pour diviser la représentation binaire de la clé privée en u blocs binaires de k bits chacun.

Selon un mode de réalisation, la primitive est une exponentiation modulaire de la donnée d'entrée par la clé privée pour la réalisation d'un algorithme de cryptographie de type RSA ou RSA CRT.

5 Selon un mode de réalisation, la primitive est une multiplication scalaire de la donnée d'entrée par la clé privée, pour la réalisation d'un algorithme de cryptographie basé sur une courbe elliptique dans lequel la donnée d'entrée est un point prédéterminé de la courbe elliptique.

10 Selon un mode de réalisation, le dispositif à microcircuit est configuré pour en outre générer initialement, de façon reproductible, au moins un paramètre de vérification avant toute exécution de la primitive, régénérer ce paramètre de vérification en cours d'exécution ou après exécution de la primitive et comparer le paramètre de vérification régénéré au paramètre de vérification généré initialement.

15 Selon un mode de réalisation, le générateur de données est configuré pour générer le paramètre de protection et/ou le paramètre de vérification en :

- définissant une fonction génératrice, par applications successives à au moins un paramètre secret prédéterminé et stocké en mémoire, d'une séquence de valeurs déterminable uniquement à partir de ce paramètre secret et de cette fonction, et

20

- générant le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette séquence.

25 Selon un mode de réalisation, le générateur de données est configuré pour :

- définir une pluralité de fonctions, chaque fonction étant génératrice, par applications successives à au moins un paramètre secret correspondant prédéterminé et stocké en mémoire, d'une séquence de valeurs correspondante déterminable uniquement à partir du paramètre secret correspondant et de la fonction correspondante,

30

- combiner la pluralité de séquences de valeurs générées à l'aide d'une relation prédéfinie pour générer une nouvelle séquence de valeurs,

- générer le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence.

35

Selon un mode de réalisation, le générateur de données est configuré pour :

- définir une fonction génératrice, par applications successives à au moins un paramètre secret prédéterminé et stocké en mémoire, d'une séquence de valeurs déterminable uniquement à partir du paramètre secret et de la fonction,

5 - combiner la séquence de valeurs générées avec des paramètres publics de l'algorithme de cryptographie pour générer une nouvelle séquence de valeurs,

10 - générer le paramètre de protection et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence.

Un autre mode de réalisation de l'invention consiste à fournir un dispositif portable, notamment une carte à puce, comprenant un dispositif à microcircuit tel que décrit précédemment.

15 Ces objets, caractéristiques et avantages ainsi que d'autres de la présente invention seront exposés plus en détail dans la description suivante, faite à titre non limitatif en relation avec les figures jointes parmi lesquelles :

- la figure 1 précédemment décrite représente de façon schématique la structure d'un dispositif à microcircuit, de type classique,

20 - la figure 2 représente de façon schématique la structure d'un dispositif à microcircuit, selon un premier mode de réalisation de l'invention,

- la figure 3 représente de façon schématique une carte à puce comprenant le dispositif de la figure 2,

25 - la figure 4 illustre les étapes successives d'un premier procédé de contre-mesure mis en œuvre par le dispositif de la figure 2,

- la figure 5 illustre les étapes successives d'un deuxième procédé de contre-mesure mis en œuvre par le dispositif de la figure 2,

- la figure 6 illustre les étapes successives d'un troisième procédé de contre-mesure mis en œuvre par le dispositif de la figure 2,

30 - la figure 7 illustre les étapes successives d'un quatrième procédé de contre-mesure mis en œuvre par le dispositif de la figure 2,

- la figure 8 illustre les étapes successives d'un cinquième procédé de contre-mesure mis en œuvre par le dispositif de la figure 2,

35 - la figure 9 représente de façon schématique la structure d'un dispositif à microcircuit, selon un deuxième mode de réalisation de l'invention, et

- la figure 10 illustre les étapes successives d'un procédé de contre-mesure mis en œuvre par le dispositif de la figure 9.

Premier mode de réalisation de l'invention.

Le dispositif à microcircuit 12' représenté sur la figure 2 comporte, comme celui représenté sur la figure 1, une application algorithmique de cryptographie asymétrique 10, une mémoire 14 incluant un espace de mémoire sécurisé 16 pour le stockage, notamment, d'une clé privée d destinée à être utilisée par l'application 10, un microprocesseur 18 et un générateur de données pseudo aléatoires 20 pour la fourniture d'un paramètre de protection a. Il comporte également une section de contre-mesure 22', mais celle-ci apporte une amélioration aux contre-mesures existantes, en particulier à la section de contre-mesure 22 précédemment décrite.

Par ailleurs, le dispositif 12' est par exemple intégré dans un dispositif portable, notamment sous la forme d'une puce de carte à puce sécurisée 30, comme représenté sur la figure 3.

On notera que, bien que l'application algorithmique de cryptographie 10 et la section de contre-mesure 22' aient été représentées comme distinctes, celles-ci peuvent être en fait intimement imbriquées en une même implémentation, logicielle ou matérielle, d'un algorithme de cryptographie asymétrique incluant une contre-mesure.

Contrairement au dispositif 12, dans ce dispositif 12' la section de contre-mesure 22' comporte :

- une section 22'a pour diviser la représentation binaire de la clé privée d en plusieurs blocs binaires $D_{u-1}, \dots, D_0$, dont la somme des tailles est par exemple égale à la taille de la représentation binaire de la clé privée ; la représentation binaire de la clé privée d peut donc s'écrire $d_{bin} = [D_{u-1}, \dots, D_0]_2$, et

- une section 22'b pour transformer chaque bloc binaire D_i à l'aide du paramètre de protection a et, pour chaque bloc binaire transformé D'_i , effectuer un calcul intermédiaire à l'aide de la primitive.

Plus précisément, le générateur 20 peut être conçu pour générer un paramètre de protection a dont la taille de la représentation binaire est au plus égale à la moitié de la taille de la représentation binaire de la clé privée d. De même, la section 22'a peut être conçue pour diviser la représentation binaire de la clé privée de telle sorte que la taille de chaque bloc binaire soit supérieure ou égale à celle de la représentation binaire du paramètre de

protection. L'application algorithmique de cryptographie asymétrique 10 exécute alors la primitive à l'aide de données dont la taille n'excède pas la moitié de celle de d_{bin} . Le gain en temps de calcul est très sensible.

Différents procédés de contre-mesure conformes à l'invention peuvent
5 être mis en œuvre par le dispositif de la figure 2.

Un premier procédé de ce type, réalisant une cryptographie de type RSA de module N sur un message M, est illustré par la figure 4. De façon classique, l'algorithme RSA nécessite l'utilisation d'une clé privée d dont la taille n de la représentation binaire est par exemple égale à $n = 1024$ bits. Si
10 l'on note d_i les bits de cette représentation binaire, il vient $d_{\text{bin}} = [d_{n-1}, \dots, d_0]_2$.

On note $S = \text{Exp}(M, D, N, S)$, la primitive suivante :

Pour i variant de j-1 à 0 :

$S \leftarrow S^2 \text{ mod } N$

si $D_i = 1$, $S \leftarrow S.M \text{ mod } N$

15 *Retourner la valeur S*

où M et S sont respectivement les données d'entrée et de sortie de la primitive, N est le module RSA et D est un exposant binaire de taille j tel que $D = [D_{j-1}, \dots, D_0]_2$, où les D_i sont les valeurs binaires de D.

Lors d'une première étape 100, le générateur 20 de données pseudo
20 aléatoires génère un paramètre de protection a dont la taille k de la représentation binaire est largement inférieure à n, par exemple $k = 32$ bits.

Lors d'une deuxième étape 102 optionnelle, on génère un paramètre de vérification r1. Ce paramètre de vérification r1 est par exemple déterminé par l'application d'une fonction COMB prédéterminée, combinant notamment
25 une valeur v générée par le générateur 20 et conservée en mémoire, le paramètre de protection a et d'autres paramètres de l'algorithme RSA.

Au cours de cette même étape optionnelle 102, on peut également transformer le message M et le module RSA N à l'aide de fonctions g et h :

$N \leftarrow h(N)$, puis

30 $M \leftarrow g(M) \text{ mod } N$,

où g et h sont par exemple des fonctions définies par $g(x) = x + r2.N$ et $h(x) = r3.x$, ou $g(x) = r2.x$ et $h(x) = x$, dans lesquelles r2 et r3 peuvent être des aléas générés par le générateur 20 et conservés en mémoire.

Ensuite, lors d'une étape d'exponentiation 104, on initialise une
35 donnée V à 1, puis on effectue le calcul suivant :

$V = \text{Exp}(M, a, N, V)$,

où V représente une donnée intermédiaire calculée à l'aide de la primitive Exp à partir de la donnée d'entrée M et du paramètre de protection a.

Lors d'une étape d'initialisation 106, on initialise la donnée de sortie S à 1 et un compteur i à n-1.

5 Ensuite, lors d'une étape de test 108, on teste la valeur du compteur i. Si cette valeur est strictement positive, on passe à une étape 110, sinon à une étape optionnelle 120 suivie d'une étape finale 122 ou directement à l'étape finale 122.

10 Lors de l'étape 110, on détermine un entier j, par exemple de façon aléatoire, qui vérifie les conditions suivantes :

(a) $k \leq j < i$, et

(b) $d_i \cdot 2^j + d_{i-1} \cdot 2^{j-1} + \dots + d_{i-j} \cdot 2^0 > a$.

En outre, si j est tel que $i-j < k$, alors on affecte à j la valeur du compteur i.

15 Ensuite, lors d'une étape 112, on calcule la valeur $D = d_i \cdot 2^j + d_{i-1} \cdot 2^{j-1} + \dots + d_{i-j} \cdot 2^0 - a$. Cette valeur D représente un bloc binaire de la clé privée d transformé par a. Puis lors d'une étape 114, on effectue le calcul intermédiaire suivant, en utilisant le bloc binaire D :

$S = \text{Exp}(M, D, N, S)$.

20 Ensuite, lors d'une étape 116, on combine la valeur intermédiaire V avec la valeur de S obtenue à l'étape 114, de la façon suivante :

$S \leftarrow S \cdot V \bmod N$.

Puis on affecte au compteur i la valeur i-j lors d'une étape 118. On revient ensuite à l'étape de test 108.

25 L'étape 120, qui est optionnelle, fait suite à l'étape 108 lorsque la valeur du compteur i est nulle et à condition que l'étape optionnelle 102 ait été réalisée. Lors de cette étape 120, on calcule de nouveau le paramètre r1, à l'aide de la fonction COMB et des valeurs publiques et/ou conservées en mémoire utilisées par cette fonction. Si la valeur de r1 a changé entre
30 l'étape 102 et l'étape 120, cela permet de conclure qu'une attaque par injection de faute(s) a eu lieu entre ces deux étapes. Une alerte est alors transmise par l'application de cryptographie 10. Lors de l'étape 120 on démasque également la donnée de sortie S, en fonction des fonctions g et h qui ont été utilisées pour masquer la donnée d'entrée M. Suivant l'alerte
35 transmise par l'application de cryptographie 10, la transformation inverse (démasquage) réalisée avec une faute permet de contrer une attaque par injection de faute(s).

Enfin, lors d'une dernière étape 122, l'application de cryptographie 10 retourne la valeur S.

On remarque que le premier procédé décrit ci-dessus implique $n+k$ itérations d'exponentiation : k itérations lors de l'étape 104 et n itérations dans la boucle des étapes 108 à 118. Lorsque k est très inférieur à n (par exemple lorsque $k = 32$ alors que $n = 1024$), le surcoût de la contre-mesure sur l'algorithme RSA est très faible. Il est en tout cas beaucoup plus faible que celui des solutions de l'état de la technique impliquant au moins $2n$ itérations d'exponentiation.

Un deuxième procédé de contre-mesure conforme à l'invention pouvant être mis en œuvre par le dispositif de la figure 2, et réalisant également une cryptographie de type RSA de module N sur un message M , est illustré par la figure 5. Il s'agit d'une variante du premier procédé dans laquelle, la taille k du paramètre de protection a étant choisie de sorte qu'il existe un entier u tel que $n = k.u$, on fixe la valeur de j (étape 110) à k et on n'impose pas la condition (b). Le procédé de contre-mesure s'en trouve simplifié.

Les étapes 200, 202 (optionnelle) et 204 de ce deuxième procédé restent identiques aux étapes 100, 102 (optionnelle) et 104 précédemment décrites.

Ensuite, lors d'une étape d'initialisation 206, on initialise la donnée de sortie S à 1 et un compteur i à $u-1$. Lors de cette même étape, on divise la représentation binaire de la clé privée d en u blocs D_i successifs, chacun de taille k , tels que $d_{\text{bin}} = [D_{u-1}, \dots, D_0]_2$. Il vient, pour tout i , $0 \leq i < u$: $D_i = [d_{k(i+1)-1}, \dots, d_{ki}]_2$. En outre, on calcule, puis on conserve en mémoire un vecteur C de retenues binaires $C = [C_{u-1}, \dots, C_0]_2$, calculé par récurrence de la façon suivante :

- $C_0 = 0$,
- $C_i = (D_i - a - C_{i-1})/2^k$.

Ensuite, lors d'une étape de test 208, on teste la valeur du compteur i . Si cette valeur est strictement positive, on passe à une étape 210, sinon à une étape optionnelle 218 suivie d'une étape finale 220 ou directement à l'étape finale 220.

Lors de l'étape 210, on calcule la valeur $D'_i = D_i - a - C_i$. Pour le bon fonctionnement de l'algorithme, si $i=u-1$ et si $C_{u-1}=1$, alors cela signifie que D'_i est inférieur à a et dans ce cas on conserve $D'_i = D_i$. Cette valeur D'_i représente le i -ème bloc binaire de la clé privée d transformé par a . On

remarque que l'un des intérêts de ce deuxième procédé est de ne nécessiter que le stockage du vecteur de retenues binaires C, et pas celui des blocs transformés D'_i .

Ensuite, lors d'une étape 212, on effectue le calcul intermédiaire
5 suivant, en utilisant le bloc binaire D'_i :

$$S = \text{Exp}(M, D'_i, N, S).$$

Ensuite, lors d'une étape 214, on combine la valeur intermédiaire V avec la valeur de S obtenue à l'étape 212, de la façon suivante :

$$S \leftarrow S.V \bmod N.$$

10 Puis on affecte au compteur i la valeur i-1 lors d'une étape 216. On revient ensuite à l'étape de test 208.

Les étapes 218 et 220 sont identiques aux étapes 120 et 122 précédemment décrites.

On remarque également que le deuxième procédé décrit ci-dessus
15 implique $n+k$ itérations d'exponentiation.

Un troisième procédé de contre-mesure conforme à l'invention pouvant être mis en œuvre par le dispositif de la figure 2, réalisant une cryptographie de type RSA CRT (ie. Algorithme RSA utilisant le Théorème du Reste Chinois) de module $N = p.q$ sur un message M, est illustré par la
20 figure 6. De façon classique, l'algorithme RSA CRT constitue une alternative à l'algorithme RSA pour effectuer une signature ou un déchiffrement : il est quatre fois plus rapide. Il définit les paramètres suivants :

- $dp = d \bmod (p-1)$,
- $dq = d \bmod (q-1)$,
- 25 - $A = p^{-1} \bmod q$.

Il consiste ensuite à remplacer le calcul d'exponentiation $S = M^d \bmod N$ par deux calculs d'exponentiation beaucoup plus simples à exécuter du fait de la taille de p et q par rapport à celle de N : $S_p = M^{dp} \bmod p$ et $S_q = M^{dq} \bmod q$. Enfin on retrouve S par le calcul suivant :

30
$$S = [((S_q - S_p).A \bmod q).p + S_p] \bmod N.$$

Les étapes 300 et 302 (optionnelle) de ce troisième procédé restent identiques aux étapes 100, 200 et 102, 202 (optionnelles) précédemment décrites.

Ensuite, lors d'une étape d'exponentiation 304, on initialise une
35 donnée V_p à 1, puis on effectue le calcul suivant :

$$V_p = \text{Exp}(M, a, p, V_p),$$

où V_p représente une donnée intermédiaire calculée à l'aide de la primitive Exp à partir de la donnée d'entrée M et du paramètre de protection a .

Suite à l'étape 304, lors d'une étape 306 incluant une série d'étapes en boucle et correspondant aux étapes déjà décrites 106 à 118 ou 206 à 216 moyennant le remplacement de l'exposant d par d_p et du module N par p , on réalise le calcul $S_p = M^{d_p} \bmod p$.

Lors d'une étape d'exponentiation 308, on initialise une donnée V_q à 1, puis on effectue le calcul suivant :

$$V_q = \text{Exp}(M, a, q, V_q),$$

où V_q représente une donnée intermédiaire calculée à l'aide de la primitive Exp à partir de la donnée d'entrée M et du paramètre de protection a .

Suite à l'étape 308, lors d'une étape 310 incluant une série d'étapes en boucle et correspondant aux étapes déjà décrites 106 à 118 ou 206 à 216 moyennant le remplacement de l'exposant d par d_q et du module N par q , on réalise le calcul $S_q = M^{d_q} \bmod q$.

L'ordre dans lequel les étapes 304 à 310 sont exécutées n'est pas figé. En effet, il importe juste qu'elles soient exécutées après l'étape 302, que l'étape 304 soit exécutée avant l'étape 306 et que l'étape 308 soit exécutée avant l'étape 310. En sortie de boucles, c'est-à-dire à l'issue des étapes 306 et 310, on passe à une étape optionnelle 312 suivie d'une étape finale 314 ou directement à l'étape finale 314.

L'étape optionnelle 312 est identique à l'étape 120 et est réalisée uniquement si l'étape optionnelle 302 a été exécutée.

Lors de l'étape finale 314, l'application de cryptographie 10 calcule la valeur de S à partir de S_p et S_q comme indiqué précédemment et retourne cette valeur.

Un quatrième procédé de contre-mesure conforme à l'invention pouvant être mis en œuvre par le dispositif de la figure 2, réalisant une cryptographie de type Courbe Elliptique sur un message M , va maintenant être présenté en référence à la figure 7. De façon classique, un algorithme de cryptographie asymétrique à Courbe Elliptique, autrement appelé algorithme ECC (de l'anglais « Elliptic Curve Cryptosystem »), nécessite l'utilisation d'une clé privée d dont la taille n est nettement inférieure à celle qui est nécessaire pour l'algorithme RSA à niveau de sécurité équivalent. En général, il faut que la représentation binaire de la clé privée d soit au moins égale à $n = 160$ bits.

Dans un algorithme ECC à clé privée d , pour effectuer une signature ou un déchiffrement, « exécuter la primitive » consiste à calculer une donnée de sortie Q à partir d'une donnée d'entrée P et de la clé privée d de la façon suivante :

5 $Q = d.P$, où P et Q sont des points d'une courbe elliptique prédéterminée sur un corps fini $GF(p)$ dans lequel p est un nombre premier strictement supérieur à 3 (par exemple la courbe elliptique $y^2 = x^3 + 10x + 5$ dans le corps $GF(13)$), et l'opération « . » est une multiplication scalaire, ici du point P par le scalaire d .

10 En notant $[d_{n-1}, \dots, d_0]_2$ la représentation binaire de la clé privée d , ce calcul peut être effectué comme suit :

$Q = 0$

Pour i variant de $n-1$ à 0 :

$Q \leftarrow 2Q$

15 $\text{si } d_i = 1, Q \leftarrow Q + P$

où « $2Q$ » et « $Q + P$ » sont respectivement des opérations de doublement de point et d'addition de points dont les formules sont déterminées de façon classique, et non détaillée ici, par la courbe elliptique choisie et l'ordre p du corps $GF(p)$.

20 Pour la suite de la description, on note $S = \text{ScalarMult}(P, D, Q)$, la primitive suivante :

Pour i variant de $j-1$ à 0 :

$Q \leftarrow 2Q$

$\text{si } D_i = 1, Q \leftarrow Q + P$

25 *Retourner la valeur Q*

où P et Q sont respectivement les données d'entrée et de sortie de la primitive et D est un exposant binaire de taille j tel que $D = [D_{j-1}, \dots, D_0]_2$, où les D_i sont les valeurs binaires de D .

30 Lors d'une première étape 400, le générateur 20 de données pseudo aléatoires génère un paramètre de protection a dont la taille k de la représentation binaire est largement inférieure à n , par exemple $k = 32$ bits.

Lors d'une deuxième étape 402 optionnelle, on génère un paramètre de vérification r . Ce paramètre de vérification r est par exemple déterminé par l'application d'une fonction COMB prédéterminée, combinant notamment
35 une valeur v générée par le générateur 20 et conservée en mémoire, le paramètre de protection a et d'autres paramètres de l'algorithme ECC.

Au cours de cette même étape optionnelle 402, on peut également transformer les coordonnées P_x et P_y du point P à l'aide d'une fonction g qui s'applique sur ces coordonnées : $P \leftarrow g(P_x, P_y) \bmod N$.

Ensuite, lors d'une étape 404, on initialise une donnée V à 0, puis on effectue le calcul suivant :

$$V = \text{ScalarMult}(P, a, V),$$

où V représente une donnée intermédiaire calculée à l'aide de la primitive ScalarMult à partir de la donnée d'entrée P et du paramètre de protection a .

Lors d'une étape d'initialisation 406, on initialise la donnée de sortie Q à 0 et un compteur i à $n-1$.

Ensuite, lors d'une étape de test 408, on teste la valeur du compteur i . Si cette valeur est strictement positive, on passe à une étape 410, sinon à une étape optionnelle 420 suivie d'une étape finale 422 ou directement à l'étape finale 422.

Lors de l'étape 410, on détermine un entier j , par exemple de façon aléatoire, qui vérifie les conditions suivantes :

$$(a) \quad k \leq j < i, \text{ et}$$

$$(b) \quad d_i \cdot 2^j + d_{i-1} \cdot 2^{j-1} + \dots + d_{i-j} \cdot 2^0 > a.$$

En outre, si j est tel que $i-j < k$, alors on affecte à j la valeur du compteur i .

Ensuite, lors d'une étape 412, on calcule la valeur $D = d_i \cdot 2^j + d_{i-1} \cdot 2^{j-1} + \dots + d_{i-j} \cdot 2^0 - a$. Cette valeur D représente un bloc binaire de la clé privée d transformé par a . Puis lors d'une étape 414, on effectue le calcul intermédiaire suivant, en utilisant le bloc binaire D :

$$Q = \text{ScalarMult}(P, D, Q).$$

Ensuite, lors d'une étape 416, on combine la valeur intermédiaire V avec la valeur de Q obtenue à l'étape 414, de la façon suivante :

$$Q \leftarrow Q + V.$$

Puis on affecte au compteur i la valeur $i-j$ lors d'une étape 418. On revient ensuite à l'étape de test 408.

L'étape 420, qui est optionnelle, fait suite à l'étape 408 lorsque la valeur du compteur i est nulle et à condition que l'étape optionnelle 402 ait été réalisée. Lors de cette étape 420, on calcule de nouveau le paramètre r , à l'aide de la fonction COMB et des valeurs publiques et/ou conservées en mémoire utilisées par cette fonction. Si la valeur de r a changé entre l'étape 402 et l'étape 420, cela permet de conclure qu'une attaque par injection de faute(s) a eu lieu entre ces deux étapes. Une alerte est alors transmise par

l'application de cryptographie 10. Lors de l'étape 420 on démasque également la donnée de sortie Q, en fonction de la fonction g qui a été utilisée pour masquer la donnée d'entrée P. Suivant l'alerte transmise par l'application de cryptographie 10, la transformation inverse (démasquage) réalisée avec une faute permet de contrer une attaque par injection de

faute(s).
Enfin, lors d'une dernière étape 422, l'application de cryptographie 10 retourne la valeur Q.

On remarque que le quatrième procédé décrit ci-dessus implique $n+k$ itérations de multiplication scalaire : k itérations lors de l'étape 404 et n itérations dans la boucle des étapes 408 à 418. Lorsque k est très inférieur à n (par exemple lorsque $k = 32$ alors que $n = 160$ ou plus), le surcoût de la contre-mesure sur l'algorithme ECC est très faible. Il est en tout cas beaucoup plus faible que celui des solutions de l'état de la technique impliquant au moins $2n$ itérations de multiplication scalaire.

En variante, lors de l'étape 404, on initialise la donnée V à 0, puis on effectue le calcul suivant : $V = \text{ScalarMult}(-P, a, V)$. Dans ce cas, lors de l'étape 412 on calcule la valeur $D = d_i \cdot 2^i + d_{i-1} \cdot 2^{i-1} + \dots + d_{i-j} \cdot 2^0 + a$. Ceci constitue une autre transformation possible de la clé privée d par a.

Un cinquième procédé de contre-mesure conforme à l'invention pouvant être mis en œuvre par le dispositif de la figure 2, et réalisant également une cryptographie à Courbe Elliptique, est illustré par la figure 8. Il s'agit d'une variante du quatrième procédé dans laquelle, la taille k du paramètre de protection a étant choisie de sorte qu'il existe un entier u tel que $n = k \cdot u$, on fixe la valeur de j (étape 410) à k et on n'impose pas la condition (b). Le procédé de contre-mesure s'en trouve simplifié.

Les étapes 500, 502 (optionnelle) et 504 de ce cinquième procédé restent identiques aux étapes 400, 402 (optionnelle) et 404 précédemment décrites.

Ensuite, lors d'une étape d'initialisation 506, on initialise la donnée de sortie Q à 0 et un compteur i à u-1. Lors de cette même étape, on divise la représentation binaire de la clé privée d en u blocs D_i successifs, chacun de taille k, tels que $d_{\text{bin}} = [D_{u-1}, \dots, D_0]_2$. Il vient, pour tout i, $0 \leq i < u$: $D_i = [d_{k(i+1)-1}, \dots, d_{ki}]_2$. En outre, on calcule, puis on conserve en mémoire un vecteur C de retenues binaires $C = [C_{u-1}, \dots, C_0]_2$, calculé par récurrence de la façon suivante :

$$- C_0 = 0,$$

$$- C_i = (D_i - a - C_{i-1})/2^k.$$

Ensuite, lors d'une étape de test 508, on teste la valeur du compteur i . Si cette valeur est strictement positive, on passe à une étape 510, sinon à une étape optionnelle 518 suivie d'une étape finale 520 ou directement à l'étape finale 520.

Lors de l'étape 510, on calcule la valeur $D'_i = D_i - a - C_i$. Pour le bon fonctionnement de l'algorithme, si $i=u-1$ et si $C_{u-1}=1$, alors cela signifie que D'_i est inférieur à a et dans ce cas on conserve $D'_i = D_i$. Cette valeur D'_i représente le i -ème bloc binaire de la clé privée d transformé par a . On remarque que l'un des intérêts de ce deuxième procédé est de ne nécessiter que le stockage du vecteur de retenues binaires C , et pas celui des blocs transformés D'_i .

Ensuite, lors d'une étape 512, on effectue le calcul intermédiaire suivant, en utilisant le bloc binaire D'_i :

$$Q = \text{ScalarMult}(P, D'_i, Q).$$

Ensuite, lors d'une étape 514, on combine la valeur intermédiaire V avec la valeur de Q obtenue à l'étape 512, de la façon suivante :

$$Q \leftarrow Q + V.$$

Puis on affecte au compteur i la valeur $i-1$ lors d'une étape 516. On revient ensuite à l'étape de test 508.

Les étapes 518 et 520 sont identiques aux étapes 420 et 422 précédemment décrites.

On remarque également que le deuxième procédé décrit ci-dessus implique $n+k$ itérations de multiplication scalaire.

De même que pour le quatrième procédé, en variante, lors de l'étape 504, on initialise la donnée V à 0, puis on effectue le calcul suivant : $V = \text{ScalarMult}(-P, a, V)$. Dans ce cas, lors de l'étape 506, le calcul du vecteur de retenues binaires est modifié de la façon suivante :

$$\begin{aligned} - C_0 &= 0, \\ - C_i &= (D_i + a + C_{i-1})/2^k. \end{aligned}$$

Dans ce cas également, lors de l'étape 510 on calcule la valeur $D'_i = D_i + a + C_i$. Ceci constitue une autre transformation possible de la clé privée d par a .

Second mode de réalisation de l'invention.

Le dispositif à microcircuit 12'' représenté sur la figure 9 comporte, comme celui représenté sur la figure 2, une application algorithmique de cryptographie 10, une mémoire 14 incluant un espace de mémoire sécurisé

16, un microprocesseur 18 et une section de contre-mesure 22'. Il est par exemple intégré dans un dispositif portable, notamment sous la forme de puce d'une carte à puce sécurisée 30 comme représenté sur la figure 3. On notera cependant que, bien que l'application algorithmique de cryptographie 10 et la section de contre-mesure 22' aient été représentées comme distinctes, celles-ci peuvent être en fait intimement imbriquées en une même implémentation d'un algorithme de cryptographie incluant une contre-mesure.

La section de contre-mesure 22' du dispositif 12'' comporte, comme celle du dispositif 12' :

- une section 22'a pour diviser la représentation binaire de la clé privée d en plusieurs blocs binaires $D_{u-1}, \dots, D_0$, dont la somme des tailles est par exemple égale à la taille de la représentation binaire de la clé privée, et
- une section 22'b pour transformer chaque bloc binaire D_i à l'aide d'un paramètre de protection a et, pour chaque bloc binaire transformé D'_i , effectuer un calcul intermédiaire à l'aide de la primitive.

Contrairement au dispositif 12', dans ce dispositif 12'' le générateur de données pseudo aléatoires 20 de type classique est remplacé par un générateur de données 20'' qui comporte :

- une section 20''a d'application d'une fonction F prédéfinie à au moins un paramètre secret prédéterminé S pour la génération d'une séquence de valeurs déterminable uniquement à partir de ce paramètre secret et de cette fonction F , et
- une section 20''b de fourniture d'au moins un paramètre de protection a de façon reproductible à partir d'une valeur de cette séquence.

La section 20''a est en fait une implémentation logicielle ou matérielle de la fonction F .

Le paramètre secret S est stocké dans la mémoire sécurisée 16 et fourni en entrée de la section 20''a du générateur 20'', tandis que le paramètre de protection a est fourni, en sortie de la section 20''b, à la section de contre-mesure 22'.

Dans ce second mode de réalisation, le paramètre a n'est donc pas un aléa au sens classique mentionné dans les documents de l'état de la technique. Il s'agit d'un résultat déterministe issu du calcul de la fonction F exécuté par le générateur 20'' sur au moins un paramètre secret S qui peut être propre à la carte à puce 30 sur laquelle est disposé le microcircuit 12'.

Ce paramètre secret est par exemple dérivé d'une donnée publique du dispositif 30.

L'application répétée de la fonction F à S génère une séquence (A_n) dont les éléments sont à l'origine du(des) paramètre(s) de protection
 5 fourni(s) par le générateur. D'une façon générale, le générateur peut fournir autant de paramètres a issus de valeurs de la séquence (A_n) que nécessaire en fonction de l'application de contre-mesure implémentée dans la carte 30. Cette séquence (A_n) ne peut être reproduite qu'avec la connaissance de la fonction génératrice F et des éléments déterministes initiaux qu'elle utilise (le
 10 paramètre S).

Chaque paramètre de protection a peut être directement issu d'un élément A_n de la séquence (A_n) : en d'autres termes, $a = A_n$. De façon alternative, l'élément A_n peut subir un traitement avant de fournir le paramètre a . Par exemple a peut être le résultat d'un calcul $a = A_n \text{ XOR } k_n$,
 15 où k_n est une constante secrète de transformation.

Bien entendu, si la séquence (A_n) est cyclique et/ou opère dans un ensemble fini d'éléments, l'espace des valeurs A_n générées devra être suffisamment grand pour résister aux attaques. En effet plus l'espace considéré est grand, meilleure est la robustesse de la contre-mesure.

20 Nous allons dans un premier temps présenter plusieurs exemples non limitatifs de séquences de valeurs (A_n) pouvant être fournies par un générateur 20'' selon le deuxième mode de réalisation de l'invention. Dans un deuxième temps, nous exposerons plusieurs utilisations possibles de telles séquences de valeurs pour la fourniture de paramètres de protection
 25 notamment aux cinq applications de contre-mesure en cryptographie asymétrique précédemment décrites en références aux figures 4 à 8.

Exemples de fonctions génératrices de séquences de valeurs pour la fourniture de paramètres de protection

1) Fonctions à base de suites arithmético-géométriques

30 Si l'on définit la séquence de valeurs (A_n) à l'aide de la fonction F entière à valeurs entières par la relation suivante :

$$A_{n+1} = F(A_n) = q.A_n + r,$$

où q et r sont des paramètres secrets constituant, avec l'élément initial A_0 de la séquence, les paramètres secrets S précédemment cités, on est capable
 35 de fournir des paramètres de protection issus d'une suite arithmético-géométrique. Les paramètres de protection sont par exemple les éléments de la séquence (A_n) .

Si $r = 0$, il s'agit d'une séquence géométrique dont on peut retrouver un terme A_i , utilisé à une étape précise de la cryptographie, à l'aide des paramètres secrets q et A_0 de la façon suivante : $A_i = q^i \cdot A_0$.

Si $q = 1$, il s'agit d'une séquence arithmétique dont on peut retrouver un terme A_i à l'aide des paramètres secrets r et A_0 de la façon suivante : $A_i = r \cdot i + A_0$.

Si r est non nul et q différent de 1, il s'agit d'une séquence arithmético-géométrique dont on peut retrouver un terme A_i à l'aide des paramètres secrets q , r et A_0 de la façon suivante :

$$A_i = q^i \cdot A_0 + r \cdot (q^i - 1) / (q - 1).$$

On peut aussi réduire l'espace des éléments de la séquence (A_n) par un nombre entier m à l'aide de la relation suivante :

$$A_{n+1} = F(A_n) \text{ modulo } m = (q \cdot A_n + r) \text{ modulo } m.$$

On remarque que si m est un nombre premier, cette séquence prend la forme du groupe des transformations affines inversibles sur le corps fini $GF(m) = \{0, 1, \dots, m-1\}$.

On peut aussi choisir m comme une puissance de 2, pour générer des séquences d'éléments à nombre de bits constant. Par exemple, si l'on veut générer des séquences de paramètres A_i à k bits, on choisit $m = 2^k$.

De préférence, m fait partie des paramètres secrets à conserver dans la mémoire sécurisée du dispositif.

2) Fonctions définissant un groupe multiplicatif cyclique

Soit un groupe cyclique GC à m éléments avec une valeur a comme élément générateur et la multiplication comme loi de composition interne : $GC = \{a, a^2, \dots, a^m\}$. On peut définir la séquence de valeurs (A_n) de la façon suivante :

- l'élément initial A_0 est choisi comme étant l'élément générateur a auquel on applique k fois la loi de composition interne du groupe GC ,
- on passe de l'élément A_i à l'élément A_{i+1} en appliquant k' fois la loi de composition interne du groupe GC .

Les paramètres secrets S utilisés par la fonction génératrice de la séquence (A_n) sont alors par exemple l'élément générateur a et les valeurs k , k' et m . De plus, comme précédemment, Les paramètres de protection générés sont par exemple les éléments de la séquence (A_n) .

3) Fonctions définissant un groupe de Frobenius

Soit un corps fini $GF(q)$, où l'ordre q est un nombre premier de k bits. Le groupe des transformations affines inversibles sur ce corps fini est un

groupe de Frobenius. Une propriété intéressante des groupes de Frobenius est qu'aucun élément non trivial ne fixe plus d'un point.

Dans ce contexte, les transformations affines utilisables prennent la forme de fonctions $y = f(x) = b.x + c$, où $b \neq 0$ et où les opérations se font dans le corps $GF(q)$. Il est donc possible de définir une fonction génératrice de la séquence (A_n) s'appliquant à des paramètres secrets q , b , c et A_0 prédéterminés. En choisissant par exemple $q = 2^{16} + 1$ et, en notation hexadécimale, $b = 0x4cd3$, $c = 0x76bb$, $A_0 = 0xef34$, on obtient une séquence commençant par les termes $A_1 = 0xc6cf$, $A_2 = 0x8baf$, $A_3 = 0x620d$, $A_4 = 0x0605$, $A_5 = 0xe70c$, $A_6 = 0x3049$, $A_7 = 0xe069$, $A_8 = 0x55ee$, etc.

4) Fonctions issues d'un registre à décalage avec rétroaction linéaire (registre de type LFSR)

Il s'agit pour ce type de fonctions de choisir un paramètre secret A_0 , par exemple de 16 bits, et un registre à décalage LFSR, par exemple avec une sortie correspondante de 16 bits. Si la taille du registre LFSR est m , alors un terme A_{t+m} de la séquence (A_n) est déterminé par les m termes qui le précèdent à l'aide d'une équation linéaire du type :

$A_{t+m} = \alpha_m.A_t + \alpha_{m-1}.A_{t+1} + \dots + \alpha_1.A_{t+m-1}$, où les α_i prennent la valeur 0 ou 1.

5) Fonctions définissant un calcul de Contrôle de Redondance Cyclique (CRC)

Il s'agit pour ce type de fonctions de choisir un paramètre secret A_0 , par exemple de 16 bits, et un polynôme CRC correspondant parmi ceux utilisés de façon classique dans les calculs de CRC, par exemple le polynôme CRC-16 ($X^{16} + X^{15} + X^2 + 1$) ou le polynôme CRC CCITT V41 ($X^{16} + X^{12} + X^5 + 1$). Un terme A_{n+1} de la séquence (A_n) est déterminé en fonction du terme précédent A_n par la relation $A_{n+1} = F(A_n)$, où F réalise un calcul de CRC sur la base du polynôme choisi.

6) Combinaisons de séquences de valeurs

Il est en effet aussi possible de calculer plusieurs séquences de valeurs, chacune par exemple selon l'une des méthodes exposées ci-dessus, et de les combiner à l'aide d'une fonction prédéfinie pour générer une nouvelle séquence de valeurs à utiliser comme paramètres de protection. On génère ainsi la séquence (A_n) , d'après deux autres séquences (A'_n) et (A''_n) , en calculant pour chaque indice n , $A_n = T(A'_n, A''_n)$.

La fonction T en question peut être une matrice secrète de valeurs, les valeurs A'_n et A''_n désignant alors respectivement une ligne et une colonne de cette matrice.

5 7) Combinaisons impliquant une séquence de valeurs et des données publiques

La séquence (A_n) peut être générée à partir d'une première séquence (A'_n), en fonction également de données publiques, telles que par exemple des données utilisées pendant l'exécution de l'application de cryptographie avec contre-mesure et non secrètes. Parmi ces données, selon les applications, on peut citer le message M (en clair ou crypté), une clé publique e, etc. Les valeurs de la séquence utilisées comme paramètres de protection sont alors calculées à l'aide d'une fonction COMB quelconque combinant toutes ces données :

$$A_n = \text{COMB}(A'_n, M, e, \dots).$$

15 Un intérêt de cette combinaison est que la séquence de valeurs (A_n) peut servir, non seulement à alimenter en paramètres de protection l'application de contre-mesure de l'algorithme de cryptographie, mais aussi à détecter des attaques par injection de fautes (notamment sur les données publiques). En effet par régénération de la séquence (A'_n) à l'aide du ou des paramètre(s) secret(s), en fin d'exécution de l'algorithme de cryptographie par exemple, mais avant de faire l'opération inverse de la transformation initiale en utilisant un paramètre de protection régénéré, puis par utilisation de cette séquence (A'_n) régénérée et des données publiques telles qu'elles apparaissent en fin d'exécution, on peut vérifier si l'application de la fonction
20 COMB produit la même séquence de valeurs (A_n) ou non et donc si des données publiques ont été affectées ou pas en cours d'exécution.

Exemples d'utilisation d'une séquence de valeurs générée selon l'une des méthodes précédentes dans un procédé de contre-mesure en cryptographie asymétrique, selon le deuxième mode de réalisation de
30 l'invention

1) Principe général du second mode de réalisation

D'une façon générale, chaque fois qu'une contre-mesure algorithmique est utilisée, la génération d'aléas introduits par la contre-mesure est préconisée, comme cela a été décrit dans le premier mode de réalisation utilisant un générateur de données pseudo aléatoires 20. Comme mentionné en référence à la figure 9, cette génération d'aléas peut être remplacée par la génération non aléatoire de paramètres issus d'une ou

plusieurs séquence(s) de valeurs obtenue(s) à l'aide d'au moins un paramètre secret.

La figure 10 illustre un exemple d'étapes réalisées par un procédé selon le second mode de réalisation de la figure 9, appliqué à l'exécution
5 d'un algorithme de cryptographie asymétrique avec contre-mesure, utilisant T paramètres de protection $a_1, \dots, a_T$ par exécution, tous les paramètres de protection pouvant être extraits d'une même séquence de valeurs (A_n) générée par la section 20'a.

Lors d'une première étape INIT réalisée par le générateur 20'', un
10 compteur i est initialisé à 0. Ce compteur i est destiné à conserver en mémoire le nombre de fois que l'algorithme de cryptographie asymétrique a été exécuté depuis cette étape d'initialisation INIT, tant qu'une autre initialisation n'est pas réalisée.

Au cours de cette étape, le paramètre secret S (ou les paramètres S
15 lorsqu'il y en a plusieurs), à partir duquel la séquence de valeurs doit être générée, est défini. Il peut être conservé d'une précédente initialisation, mais peut aussi être généré sur la base d'une nouvelle valeur à l'occasion de cette initialisation. Il est par exemple généré à partir de données uniques d'identification, telle qu'une donnée publique du dispositif 30. Il peut aussi
20 être généré à partir de paramètres ou phénomènes physiques liés au microcircuit à un instant donné, qui peuvent être aléatoires. Dans tous les cas, il est conservé en mémoire de façon sécurisée, pour permettre au microcircuit de régénérer à tout moment une même séquence de valeurs (A_n) à l'aide de la fonction implémentée par la section 20''a.

25 L'étape d'initialisation INIT peut être unique dans le cycle de vie du microcircuit, réalisée lors de la conception par le constructeur, ou reproduite plusieurs fois, par exemple régulièrement ou chaque fois que le compteur i atteint une valeur imax.

Lors d'une première exécution EXE1 de l'algorithme de cryptographie
30 asymétrique avec contre-mesure, le générateur 20'', plus particulièrement la section 20''a, est sollicité une ou plusieurs fois pour appliquer le paramètre secret S à la fonction F prédéfinie, de manière à générer, en une ou plusieurs fois, un nombre T d'éléments de la séquence de valeurs (A_n) : $A_1, \dots, A_T$. A partir de ces T premiers éléments, les T paramètres de protection
35 $a_1, \dots, a_T$ sont générés.

Par exemple, pour tout k tel que $1 \leq k \leq T$, $a_k = A_k$.

En variante, si l'on dispose de T valeurs secrètes supplémentaires $Sec_1, \dots, Sec_T$ parmi les paramètres secrets S conservés en mémoire sécurisée, on peut effectuer le calcul supplémentaire suivant :

pour tout k tel que $1 \leq k \leq T$, $a_k = Sec_k \text{ XOR } A_k$, ou $a_k = Sec_k \text{ ADD } A_k$, ou
 5 bien également $a_k = Sec_k \text{ SUB } A_k$, de manière à transformer (ou déformer ou masquer) les paramètres utilisés.

Par la suite, lors d'une i-ème exécution EXEi de l'algorithme de cryptographie avec contre-mesure, le générateur 20'', plus particulièrement la section 20''a, est de nouveau sollicité une ou plusieurs fois pour appliquer
 10 le paramètre secret S à la fonction F prédéfinie, de manière à générer, en une ou plusieurs fois, un nombre T d'éléments supplémentaires de la séquence de valeurs (A_n) : $A_{T(i-1)+1}, \dots, A_{Ti}$. A partir de ces T éléments supplémentaires, les T paramètres de protection $a_1, \dots, a_T$ sont générés, comme précédemment.

15 Par exemple, pour tout k tel que $1 \leq k \leq T$, $a_k = A_{T(i-1)+k}$.

En variante, si l'on dispose des T valeurs secrètes supplémentaires $Sec_1, \dots, Sec_T$, on peut effectuer le calcul supplémentaire suivant :

pour tout k tel que $1 \leq k \leq T$, $a_k = Sec_k \text{ XOR } A_{T(i-1)+k}$, ou $a_k = Sec_k \text{ ADD } A_{T(i-1)+k}$, ou bien également $a_k = Sec_k \text{ SUB } A_{T(i-1)+k}$, de manière à transformer (ou
 20 déformer ou masquer) les paramètres utilisés.

Quelle que soit la méthode utilisée pour générer la ou les séquence(s) de valeurs à l'origine des paramètres de protection, la connaissance de la méthode et des valeurs secrètes utilisées par la méthode, y compris le paramètre initial A_0 chargé préalablement en mémoire ou lors d'une étape du
 25 cycle de vie du dispositif à microcircuit en mémoire EEPROM, permet de retrouver à tout moment les paramètres de protection générés et utilisés dans la vie du dispositif. Il apparaît clairement que cette particularité permet alors des débogages simples et efficaces ainsi qu'une résistance améliorée aux attaques par injection de fautes.

30 Le choix de la méthode utilisée pour générer la séquence de valeurs et le ou les paramètre(s) de protection est dicté par l'application envisagée.

2) Application du principe général du second mode de réalisation aux cinq procédés décrits en référence aux figures 4 à 8

La méthode utilisée par les premier, second et troisième procédés des
 35 figures 4, 5 et 6 pour générer le paramètre de protection a lors des étapes 100, 200, 300 et les paramètres v, r2, r3 lors des étapes 102, 202, 302 peut être l'une de celles préconisées dans le second mode de réalisation. En

outre, les paramètres a , v , r_2 , r_3 peuvent avoir la même taille binaire et être issus d'une même séquence de valeurs ($T=4$). Par ailleurs, il n'est pas utile de conserver ces paramètres en mémoire puisqu'ils peuvent être retrouvés à chaque instant à partir de la séquence de valeurs qui est elle même déterminée par le(s) paramètre(s) secret(s) et la fonction F . C'est ainsi que
5 les paramètres v puis r_1 , r_2 et r_3 peuvent être retrouvés aux étapes 120, 218, 312 sans avoir nécessairement été conservés en mémoire pendant l'exécution de l'exponentiation. A ces étapes 120, 218, 312, le paramètre de protection a peut également être retrouvé pour vérifier que son intégrité a été
10 conservée pendant l'exponentiation.

De même, la méthode utilisée par les quatrième et cinquième procédés des figures 7 et 8 pour générer le paramètre de protection a lors des étapes 400, 500 et le paramètre v lors des étapes 402, 502 peut être l'une de celles préconisées dans le second mode de réalisation. En outre,
15 les paramètres a et v peuvent avoir la même taille binaire et être issus d'une même séquence de valeurs ($T=2$). Par ailleurs, il n'est pas utile de conserver ces paramètres en mémoire puisqu'ils peuvent être retrouvés à chaque instant à partir de la séquence de valeurs qui est elle même déterminée par le(s) paramètre(s) secret(s) et la fonction F . Ce processus consistant à
20 régénérer ces paramètres est même une étape utile dans la protection de l'implémentation contre les attaques par injection de faute(s). C'est ainsi que le paramètre v puis r peut être retrouvé aux étapes 420, 518 sans avoir nécessairement été conservé en mémoire pendant l'exécution de la multiplication scalaire. A ces étapes 420, 518, le paramètre de protection a
25 peut également être retrouvé pour vérifier que son intégrité, et celle des paramètres utilisés pour le générer, a été conservée pendant la multiplication scalaire.

Une protection supplémentaire peut être ajoutée lors de l'exécution de la boucle de calcul de la primitive, dans chacun des cinq procédés précités.
30 On génère au préalable un paramètre de vérification s selon l'une des méthodes préconisées ci-dessus, ce paramètre venant s'ajouter aux paramètres a et v , r_1 ou a , v , r_1 , r_2 et r_3 . A chaque itération dans cette boucle de calcul, par exemple à l'étape 118 du premier procédé, à l'étape 216 du deuxième procédé, aux étapes 306 et 310 du troisième procédé, à
35 l'étape 418 du quatrième procédé et à l'étape 516 du cinquième procédé, on retrouve s et on extrait de façon déterministe à l'aide du paramètre s des portions d'au moins une partie des représentations binaires ou de

représentations selon une autre base b du message M , du module N (dans le cas de RSA ou RSA CRT), de la clé privée d , etc. Ces portions sont alors notées M_s , N_s , d_s , etc. et éventuellement combinées pour former une donnée de vérification. Le principe de cette protection est de vérifier qu'à
5 chaque itération, la valeur de la donnée de vérification est inchangée. Si cette donnée de vérification change, les données M , N , d , etc. peuvent être brouillées pour ne pas être découvertes et une alerte peut être déclenchée. D'autres données que M , N et d peuvent être utilisées, à partir du moment où ces données sont utilisées lors de l'exécution de la primitive.

10 Il apparaît clairement que les procédés de contre-mesure décrits précédemment permettent de concevoir des applications de cryptographie asymétriques protégeant la clé privée utilisée contre des attaques par canaux auxiliaires, tout en limitant le surcoût en temps de calcul à un niveau très raisonnable.

15 On notera en outre que l'invention n'est pas limitée aux modes de réalisation décrits et que, bien que de nombreuses variantes aient été présentées, d'autres sont également envisageables prévoyant notamment d'autres types de transformations de la clé privée que ceux qui ont été détaillés, ou d'autres applications de cryptographie asymétrique que celles
20 abordées.

REVENDICATIONS

1. Procédé de contre-mesure dans un composant
5 électronique mettant en œuvre un algorithme de
cryptographie asymétrique à clé privée (d), comprenant
les étapes consistant à :
- générer (100 ; 200 ; 300 ; 400 ; 500) un paramètre
de protection (a),
 - 10 - calculer (104 ; 204 ; 304, 308 ; 404 ; 504), à
l'aide d'une primitive de l'algorithme de cryptographie,
une donnée intermédiaire à partir d'une donnée d'entrée
et du paramètre de protection (a),
caractérisé en ce qu'il comporte en outre les étapes
15 consistant à :
 - diviser (110 ; 206 ; 306, 310 ; 410 ; 506) la
représentation binaire de la clé privée (d) en plusieurs
blocs binaires,
 - transformer (112 ; 210 ; 306, 310 ; 412 ; 510)
20 chaque bloc binaire à l'aide du paramètre de protection
(a) et, pour chaque bloc binaire transformé, effectuer
(114 ; 212 ; 306, 310 ; 414 ; 512) un calcul
intermédiaire à l'aide de la primitive, et
 - calculer (106-122 ; 206-220 ; 306, 310, 312, 314 ;
25 406-422 ; 506-520) une donnée de sortie par combinaison
(116 ; 214 ; 306, 310 ; 416 ; 514) de la donnée
intermédiaire avec les calculs intermédiaires (114 ;
212 ; 306, 310 ; 414 ; 512).
- 30 2. Procédé de contre-mesure dans un composant
électronique selon la revendication 1, comprenant l'étape
consistant à diviser (110 ; 206 ; 306, 310 ; 410 ; 506)
la représentation binaire de la clé privée de telle sorte
que la taille de chaque bloc binaire (d) soit supérieure
35 ou égale à celle de la représentation binaire du
paramètre de protection (a).

3. Procédé de contre-mesure dans un composant électronique selon la revendication 1 ou 2, comprenant l'étape consistant à diviser (110 ; 206 ; 306, 310 ; 410 ; 506) la représentation binaire de la clé privée (d) en plusieurs blocs binaires de telle sorte que la somme des tailles des blocs binaires soit égale à la taille de la représentation binaire de la clé privée.

4. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 3, comprenant l'étape consistant à déterminer aléatoirement (110 ; 410) de manière itérative la taille de chaque bloc binaire de telle sorte que la valeur de chaque bloc binaire soit supérieure à la valeur du paramètre de protection (a).

5. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 3, comprenant les étapes consistant à :

- choisir la taille k de la représentation binaire du paramètre de protection (a) de telle sorte qu'il existe un entier $u \geq 2$ tel que $n = k \cdot u$, n étant la taille de la représentation binaire de la clé privée (d), et
- diviser (206 ; 506) la représentation binaire de la clé privée en u blocs binaires de k bits chacun.

6. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 5, dans lequel la primitive est une exponentiation modulaire de la donnée d'entrée par la clé privée (d) pour la réalisation d'un algorithme de cryptographie de type RSA ou RSA CRT.

7. Procédé de contre-mesure dans un composant électronique selon la revendication 6, comprenant une étape préalable (102 ; 202 ; 302) consistant à masquer le module RSA et la donnée d'entrée.

8. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 5, dans lequel la primitive est une multiplication scalaire de la donnée d'entrée par la clé privée (d), pour la réalisation d'un algorithme de cryptographie basé sur une courbe elliptique dans lequel la donnée d'entrée est un point prédéterminé de la courbe elliptique.

10 9. Procédé de contre-mesure dans un composant électronique selon la revendication 8, comprenant une étape préalable (402 ; 502) consistant à masquer le point prédéterminé de la courbe elliptique.

15 10. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 9, comportant en outre les étapes consistant à :

- générer initialement (102 ; 202 ; 302 ; 402 ; 502), de façon reproductible, au moins un paramètre de vérification avant toute exécution de la primitive,
20 - régénérer (120, 118 ; 218, 216 ; 312, 306, 310 ; 420, 418 ; 518, 516) ce paramètre de vérification en cours d'exécution ou après exécution de la primitive et comparer le paramètre de vérification régénéré au
25 paramètre de vérification généré initialement.

11. Procédé de contre-mesure dans un composant électronique selon la revendication 10, dans lequel l'étape (118 ; 216 ; 306, 310 ; 418 ; 516) de régénération et de comparaison est exécutée à chaque itération de la primitive lorsqu'elle est appliquée à un bloc binaire transformé.

12. Procédé de contre-mesure dans un composant électronique selon la revendication 10 ou 11, comprenant l'étape consistant à déclencher une alerte et brouiller au moins la clé privée (d), si l'étape de régénération et

de comparaison indique une différence entre le paramètre de vérification généré initialement et le paramètre de vérification régénéré.

- 5 13. Procédé de contre-mesure dans un composant électronique selon l'une quelconque des revendications 1 à 12, dans lequel la génération (100, 102 ; 200, 202 ; 300, 302 ; 400, 402 ; 500, 502) du paramètre de protection et/ou du paramètre de vérification comporte
- 10 les étapes consistant à :
- prévoir au moins un paramètre secret (S) prédéterminé stocké dans une mémoire sécurisée (16) du composant électronique,
 - définir une fonction génératrice (20''a)

15 permettant de générer une séquence de valeurs ((A_n)) par applications successives de la fonction génératrice au paramètre secret (S), la séquence de valeurs ((A_n)) étant déterminable uniquement à partir de la fonction génératrice et du paramètre secret (S),

 - générer au moins une séquence de valeurs ((A_n)) au

20 moyen de la fonction génératrice et du paramètre secret, et

 - générer le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir

25 d'au moins une valeur de cette séquence de valeurs.

14. Procédé de contre-mesure dans un composant électronique selon la revendication 13, comportant les étapes consistant à :
- 30 - définir une pluralité de fonctions, chaque fonction étant génératrice, par applications successives à au moins un paramètre secret (S) correspondant prédéterminé et stocké en mémoire (16), d'une séquence de valeurs ((A'_n), (A''_n)) correspondante déterminable
- 35 uniquement à partir du paramètre secret (S) correspondant et de la fonction correspondante,

- combiner la pluralité de séquences de valeurs $((A'_n), (A''_n))$ générées à l'aide d'une relation prédéfinie pour générer une nouvelle séquence de valeurs $((A_n))$,

- 5 - générer le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence $((A_n))$.

15 15. Procédé de contre-mesure dans un composant électronique selon la revendication 13, comportant les étapes consistant à :

- définir une fonction génératrice, par applications successives à au moins un paramètre secret (S) prédéterminé et stocké en mémoire (16), d'une
15 séquence de valeurs $((A'_n))$ déterminable uniquement à partir du paramètre secret (S) et de la fonction,

- combiner la séquence de valeurs $((A'_n))$ générées avec des paramètres publics de l'algorithme de cryptographie pour générer une nouvelle séquence de
20 valeurs $((A_n))$,

- générer le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence $((A_n))$.

25 16. Dispositif à microcircuit $(12', 12'')$, comprenant un microprocesseur (18) pour la mise en œuvre d'un procédé de contre-mesure d'un algorithme de cryptographie asymétrique à clé privée (d), au moins une mémoire sécurisée (16) pour le stockage de la clé privée
30 (d), et un générateur de données $(20, 20'')$ pour la génération d'un paramètre de protection (a), caractérisé en ce qu'il est configuré pour :

- calculer $(104 ; 204 ; 304, 308 ; 404 ; 504)$, à l'aide d'une primitive de l'algorithme de cryptographie,
35 une donnée intermédiaire à partir d'une donnée d'entrée et du paramètre de protection (a),

- diviser (110 ; 206 ; 306, 310 ; 410 ; 506) la représentation binaire de la clé privée (d) en plusieurs blocs binaires,

5 - transformer (112 ; 210 ; 306, 310 ; 412 ; 510) chaque bloc binaire à l'aide du paramètre de protection (a) et, pour chaque bloc binaire transformé, effectuer (114 ; 212 ; 306, 310 ; 414 ; 512) un calcul intermédiaire à l'aide de la primitive,

10 - calculer (106-122 ; 206-220 ; 306, 310, 312, 314 ; 406-422 ; 506-520) une donnée de sortie par combinaison (116 ; 214 ; 306, 310 ; 416 ; 514) de la donnée intermédiaire avec les calculs intermédiaires (114 ; 212 ; 306, 310 ; 414 ; 512).

15 17. Dispositif à microcircuit (12', 12'') selon la revendication 16, dans lequel le microprocesseur (18) est configuré pour déterminer aléatoirement (110 ; 410) de manière itérative la taille de chaque bloc binaire de telle sorte que la valeur de chaque bloc binaire soit
20 supérieure à la valeur du paramètre de protection (d).

18. Dispositif à microcircuit (12', 12'') selon la revendications 16, dans lequel le générateur de données (20, 20'') est configuré pour choisir la taille k de la
25 représentation binaire du paramètre de protection (a) de telle sorte qu'il existe un entier $u \geq 2$ tel que $n = k \cdot u$, n étant la taille de la représentation binaire de la clé privée (d), et dans lequel le microprocesseur (18) est configuré pour diviser (206 ; 506) la représentation
30 binaire de la clé privée en u blocs binaires de k bits chacun.

19. Dispositif à microcircuit (12', 12'') selon l'une quelconque des revendications 16 à 18, dans lequel
35 la primitive est une exponentiation modulaire de la donnée d'entrée par la clé privée (d) pour la réalisation d'un algorithme de cryptographie de type RSA ou RSA CRT.

20. Dispositif à microcircuit (12', 12'') selon l'une quelconque des revendications 16 à 18, dans lequel la primitive est une multiplication scalaire de la donnée d'entrée par la clé privée (d), pour la réalisation d'un algorithme de cryptographie basé sur une courbe elliptique dans lequel la donnée d'entrée est un point prédéterminé de la courbe elliptique.

21. Dispositif à microcircuit (12', 12'') selon l'une quelconque des revendications 16 à 20, configuré pour en outre générer initialement (102 ; 202 ; 302 ; 402 ; 502), de façon reproductible, au moins un paramètre de vérification avant toute exécution de la primitive, régénérer (120, 118 ; 218, 216 ; 312, 306, 310 ; 420, 418 ; 518, 516) ce paramètre de vérification en cours d'exécution ou après exécution de la primitive et comparer le paramètre de vérification régénéré au paramètre de vérification généré initialement.

22. Dispositif à microcircuit (12', 12'') selon l'une quelconque des revendications 16 à 21, dans lequel le générateur de données (20'') est configuré pour générer (100, 102 ; 200, 202 ; 300, 302 ; 400, 402 ; 500, 502) le paramètre de protection et/ou le paramètre de vérification en :

- générant une séquence de valeurs ((A_n)) par applications successives d'une fonction génératrice (20''a) prédéfinie à au moins un paramètre secret prédéterminé (S) stocké dans la mémoire sécurisée (16), la séquence de valeurs ((A_n)) étant déterminable uniquement à partir du paramètre secret (S) et de la fonction génératrice (20''a), et

- fournissant le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette séquence de valeurs ((A_n)).

23. Dispositif à microcircuit (12', 12'') selon la revendication 22, dans lequel le générateur de données (20'') est configuré pour :

- 5 - définir une pluralité de fonctions, chaque fonction étant génératrice, par applications successives à au moins un paramètre secret (S) correspondant prédéterminé et stocké en mémoire (16), d'une séquence de valeurs ((A'_n), (A''_n)) correspondante déterminable
10 uniquement à partir du paramètre secret (S) correspondant et de la fonction correspondante,

- combiner la pluralité de séquences de valeurs ((A'_n), (A''_n)) générées à l'aide d'une relation prédéfinie pour générer une nouvelle séquence de valeurs
15 ((A_n)),

 - générer le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence ((A_n)).

20 24. Dispositif à microcircuit (12', 12'') selon la revendication 22, dans lequel le générateur de données (20'') est configuré pour :

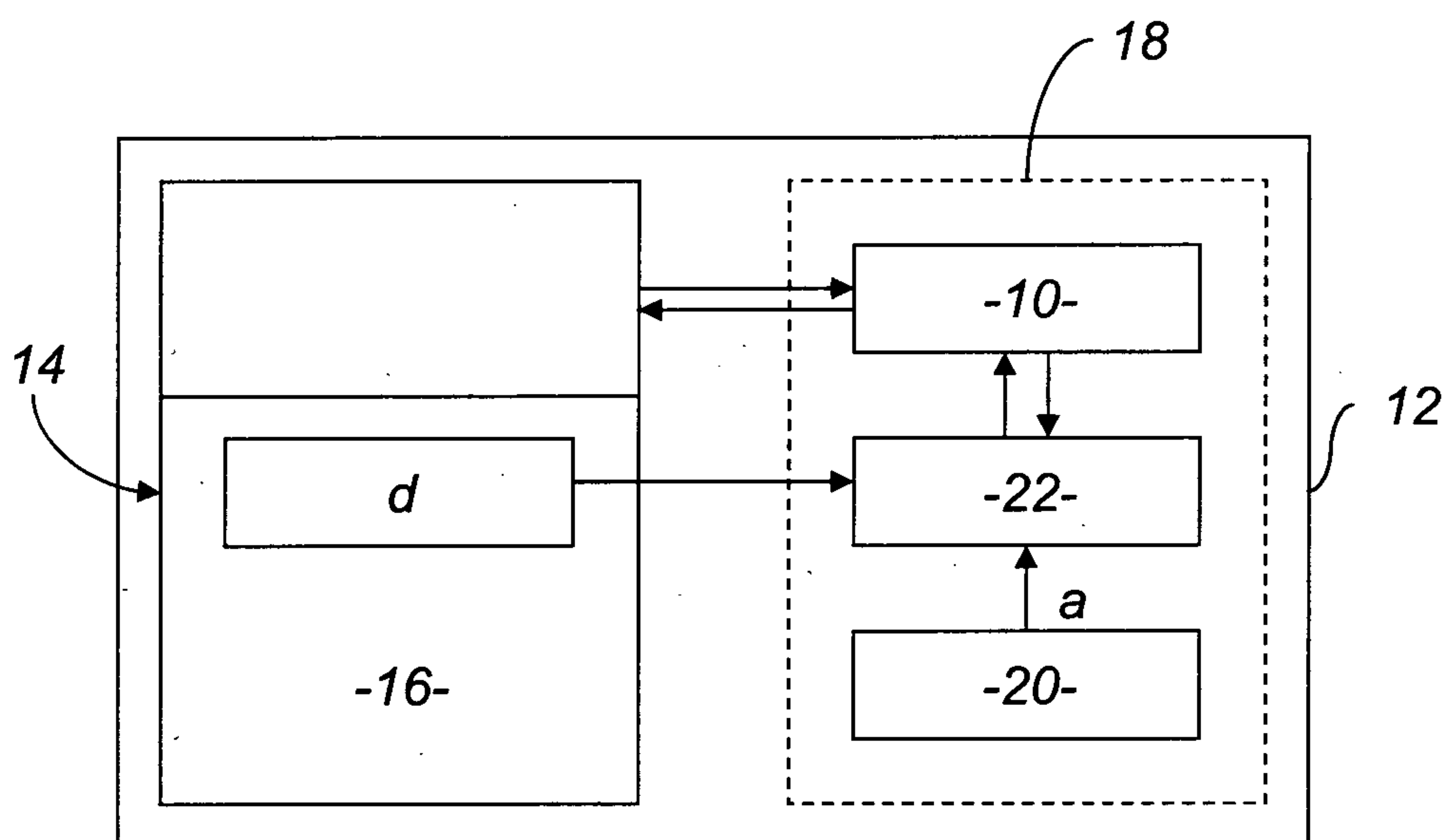
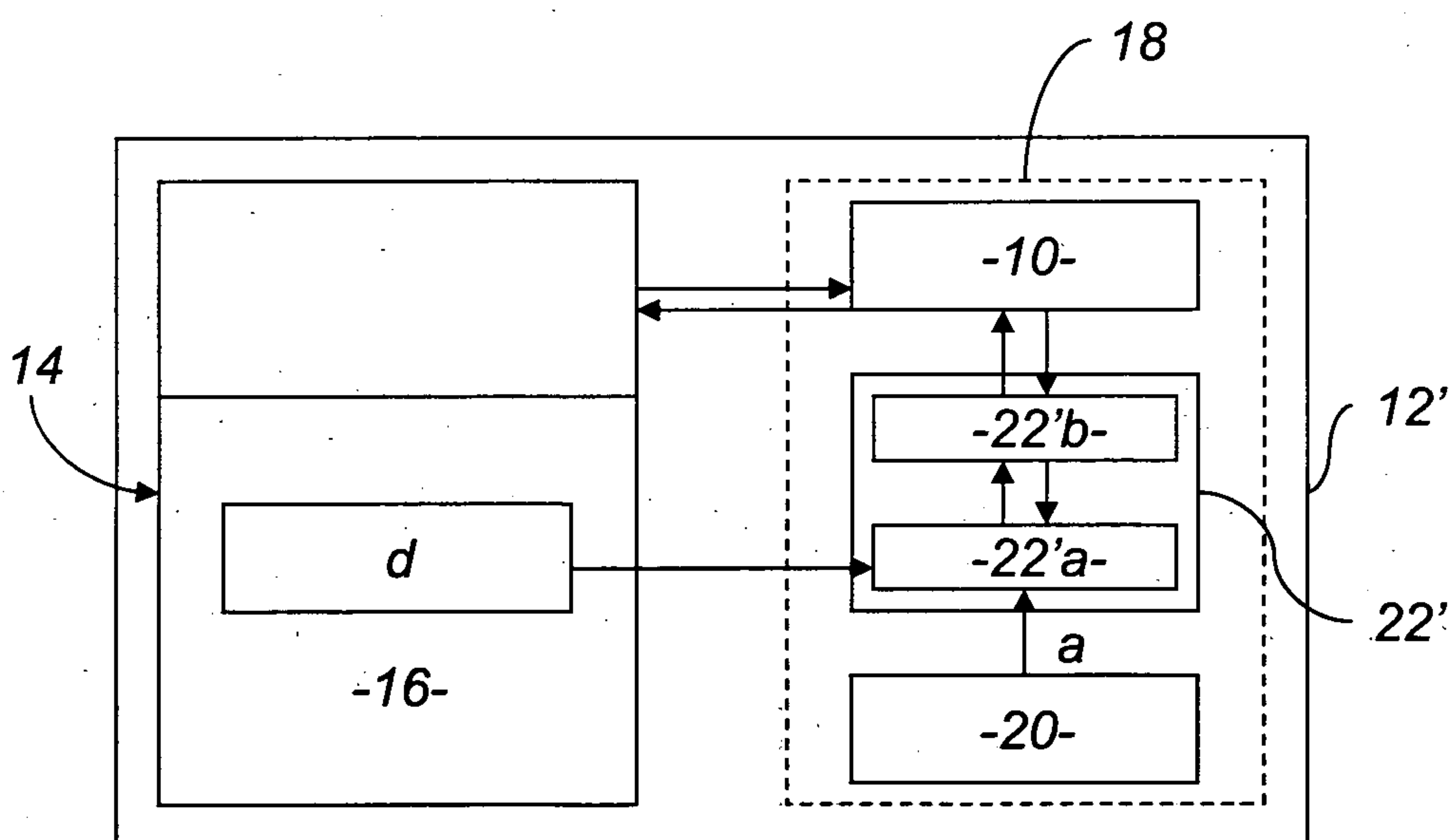
- définir une fonction génératrice, par applications successives à au moins un paramètre secret
25 (S) prédéterminé et stocké en mémoire (16), d'une séquence de valeurs ((A'_n)) déterminable uniquement à partir du paramètre secret (S) et de la fonction,

- combiner la séquence de valeurs ((A'_n)) générées avec des paramètres publics de l'algorithme de
30 cryptographie pour générer une nouvelle séquence de valeurs ((A_n)),

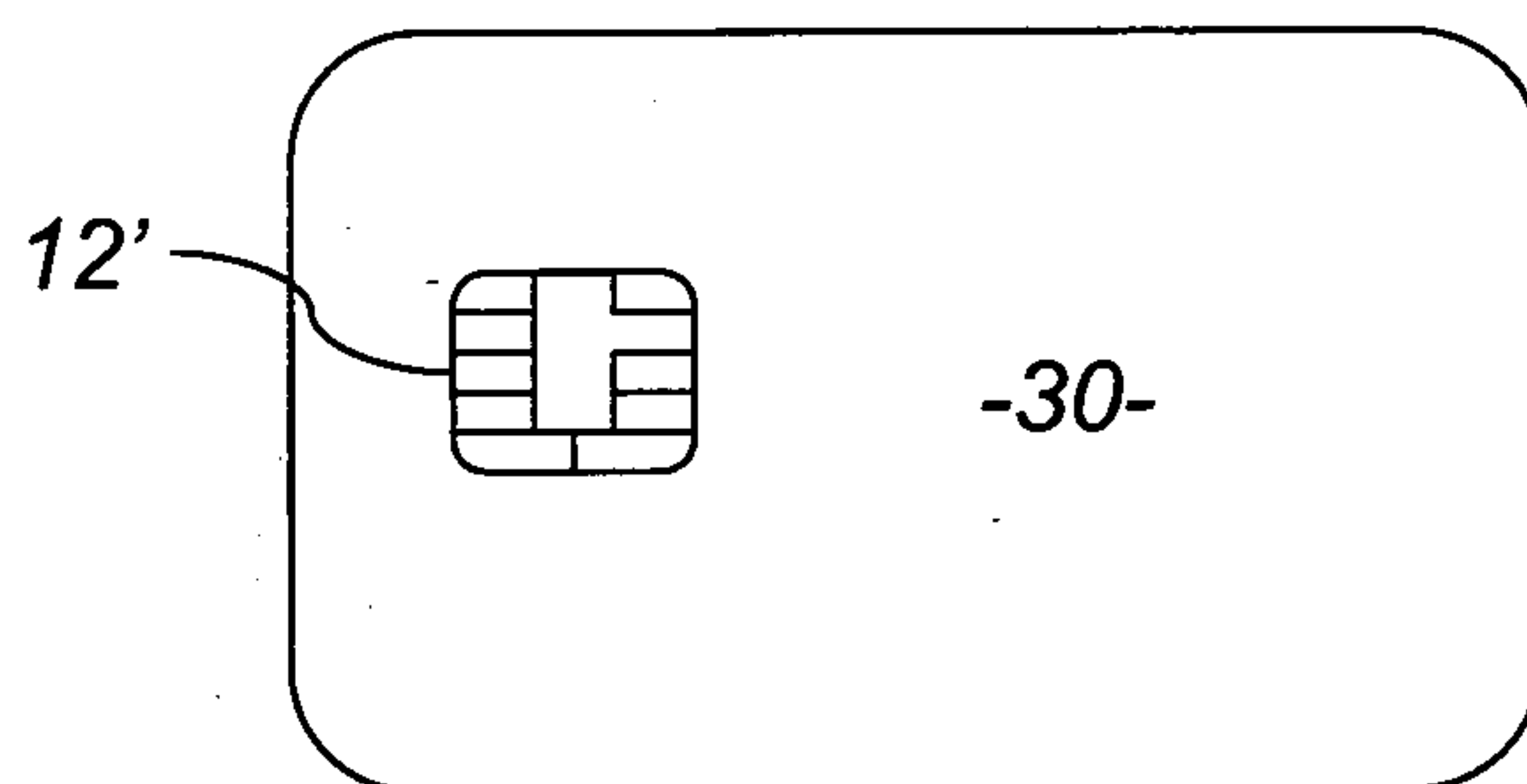
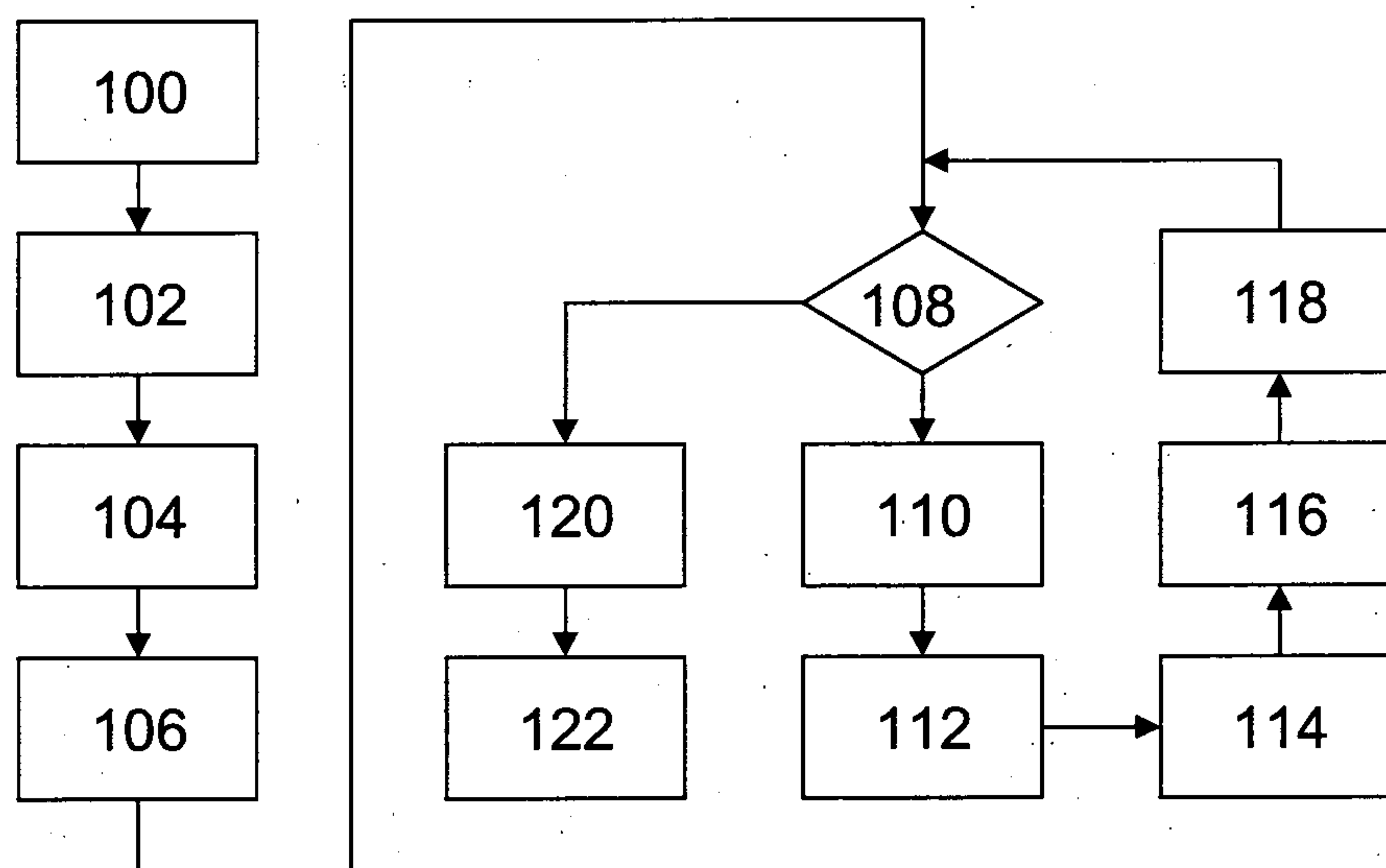
 - générer le paramètre de protection (a) et/ou le paramètre de vérification de façon reproductible à partir d'au moins une valeur de cette nouvelle séquence ((A_n)).

25. Dispositif portable, notamment carte à puce (30), comprenant un dispositif à microcircuit (12', 12'') selon l'une quelconque des revendications 16 à 24.

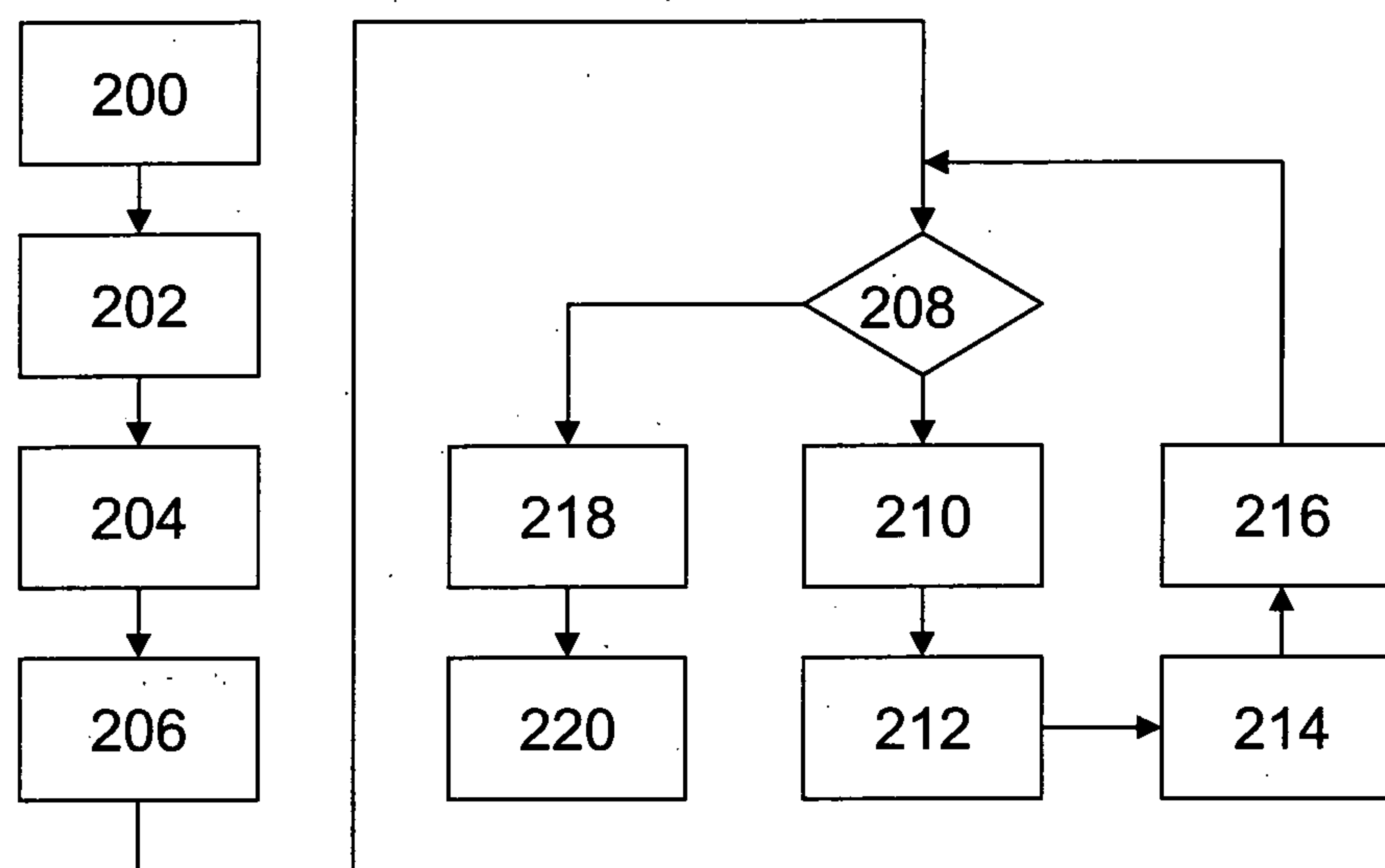
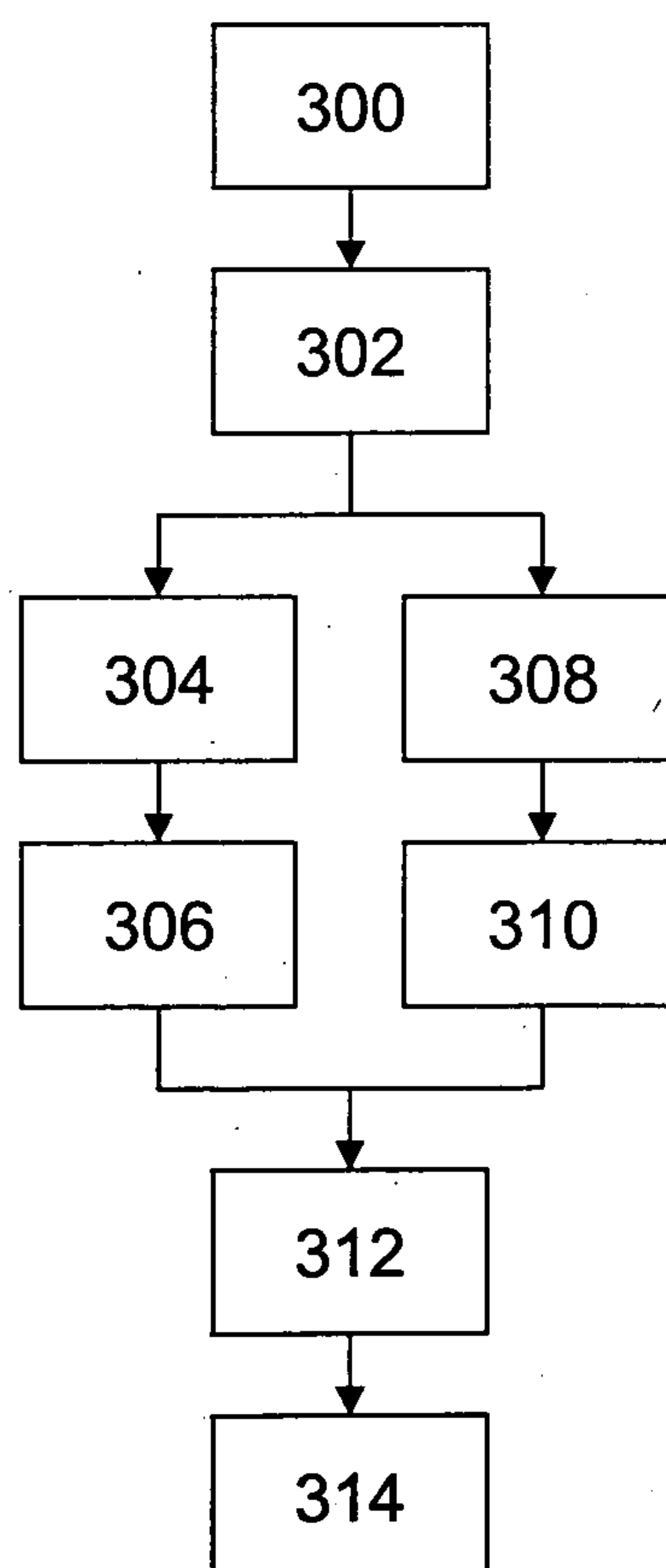
1/5

Figure 1Figure 2

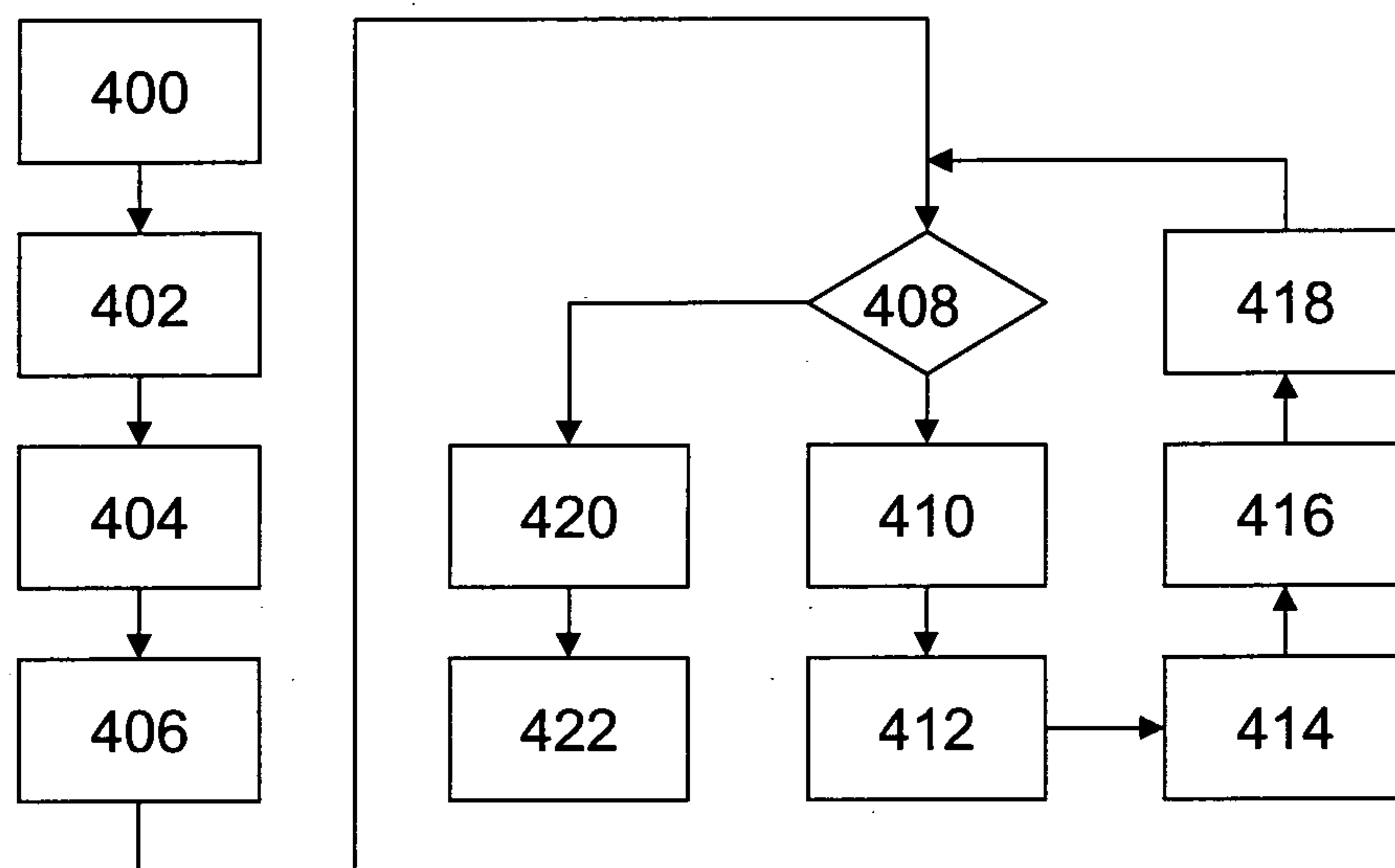
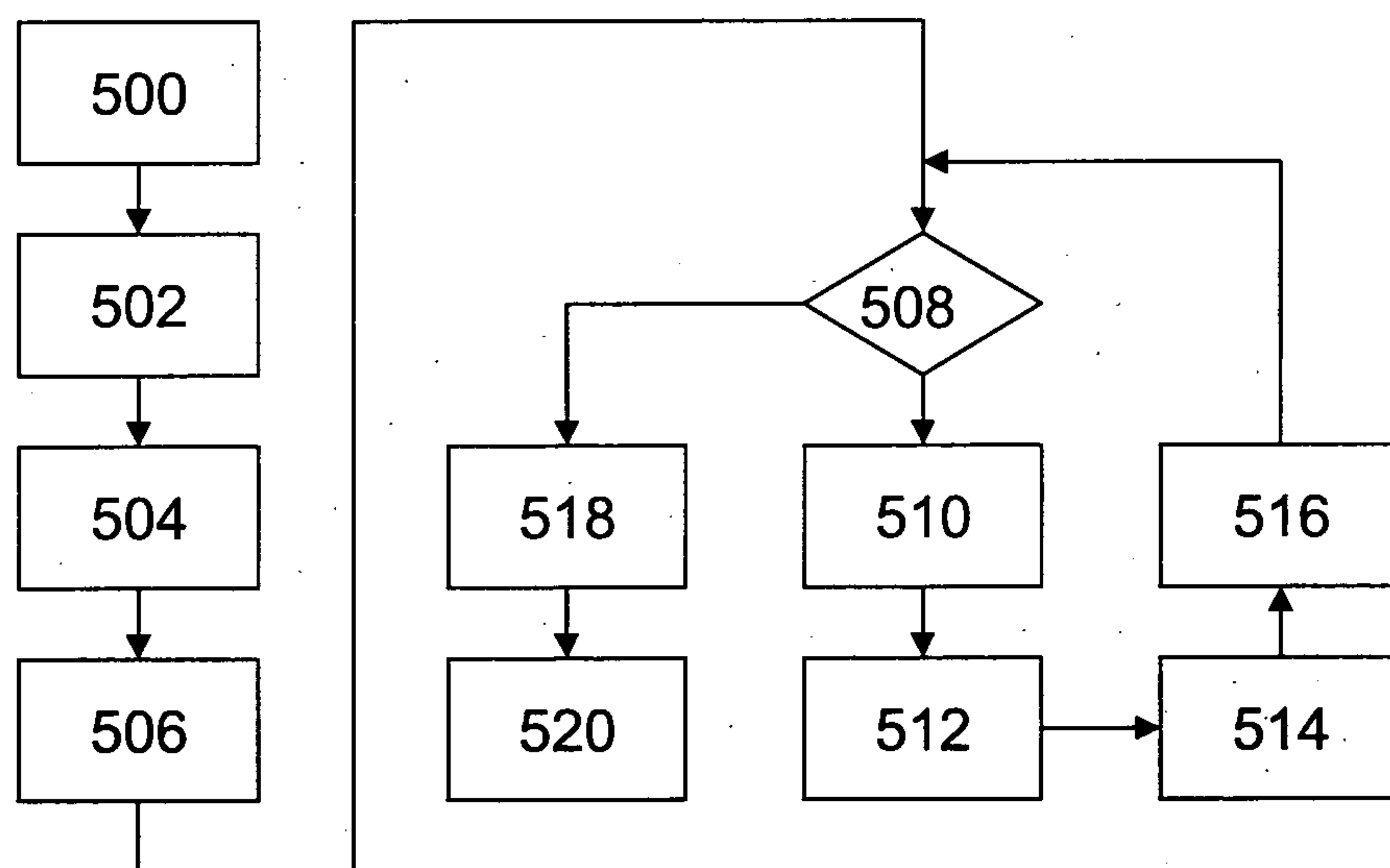
2/5

Figure 3Figure 4

3/5

Figure 5Figure 6

4/5

Figure 7Figure 8

5/5

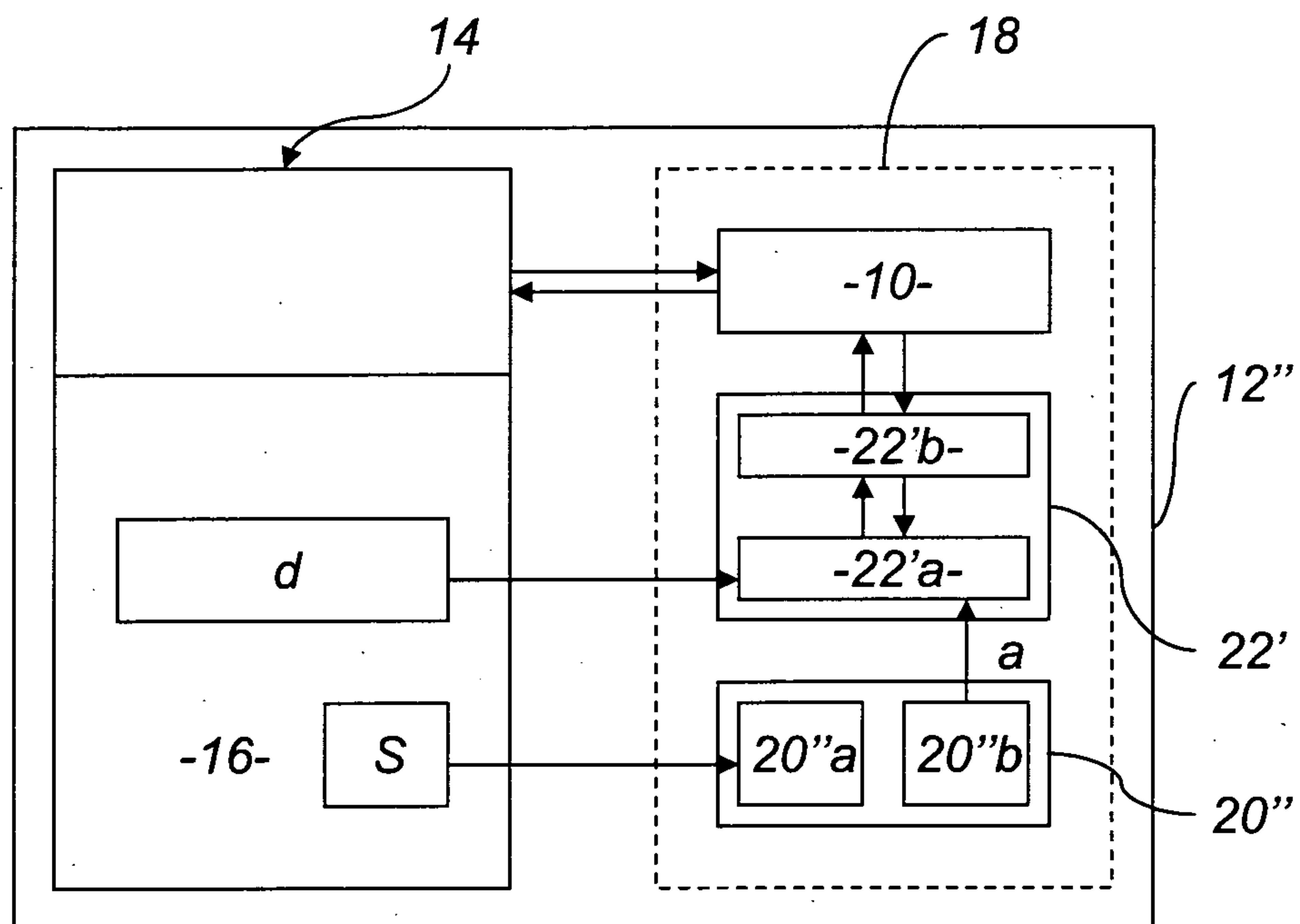
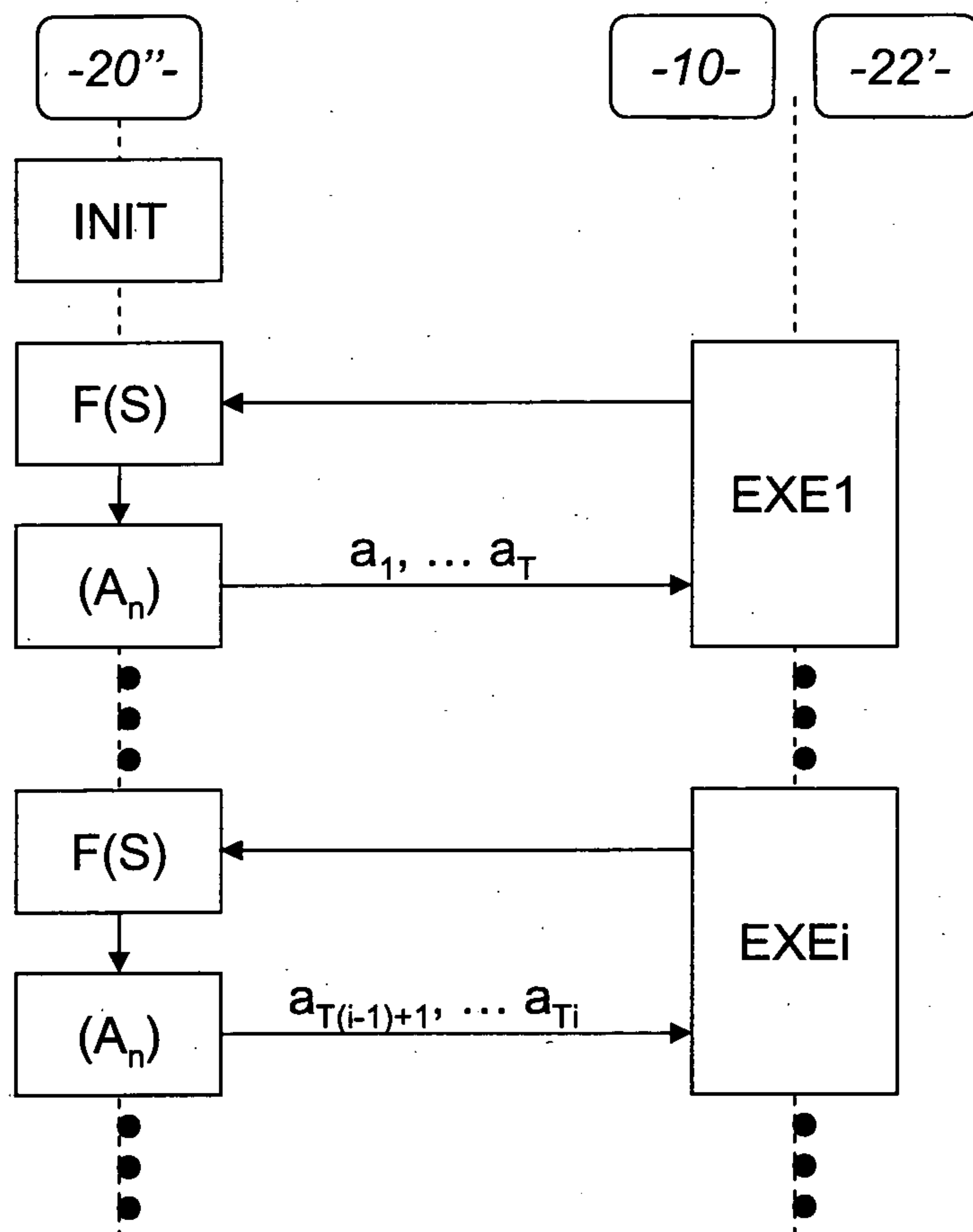
Figure 9Figure 10

Figure 4

