Silesian University of Technology
Faculty of Automatic Control, Electronics and Computer Science
Institute of Electronics
University of Rennes 1
IRISA

Résumé

Arithmetic operators on $GF(2^m)$ for cryptographic applications : performance - power consumption - security tradeoffs

(Opérateurs arithmétiques sur $GF(2^m)$: étude de compromis performances-consommation-sécurité.)

Author :
Danuta Pamuła

Supervisors :
dr hab. inż. Edward Hrynkiewicz,
prof. nzw. w Politechnice Śląskiej (PL)
Arnaud Tisserand, CNRS researcher, HDR (FR)

Gliwice 2012

Résumé

1. Introduction

De nos jours, les systèmes numériques et l'Internet sont présents dans presque tous les domaines de notre vie. Ils sont responsables de communication entre les personnes, les institutions et entre les gouvernements. Ils sont utilisés pour surveiller des systèmes de contrôle du trafic aérien, du transport, des systèmes médicaux, bancaires, des marchés financier, militaires, etc. Nous sommes inondés de données numériques, qui ne sont pas toujours faciles à identifier, autoriser, sécuriser ou stocker. La plupart des utilisateurs réguliers de systèmes numériques ne connaissent pas bien les effets de mauvaises manipulations de leurs données numériques ou de l'usurpation de leur identité numérique.

L'identité numérique n'est pas traitée par la plupart des utilisateurs de systèmes à égalité avec l'identité "réelle" représentée par une carte d'identité ou un passeport. Les effets de prise de contrôle ou de la perte des deux identités sont presque identiques, mais la mauvaise utilisation d'identité numérique peut avoir de conséquences encore plus sérieuses [21].

Heureusement, la conscience de la nécessité de protéger et de sécuriser les identités numériques et les données augmente avec le nombre de services disponibles dans le réseau mondial (Internet). Tout le monde veut faire des achats électroniques en toute sécurité sur Internet, gérer des ressources bancaires sans crainte de détournement de fonds, signer des documents numériques d'une manière propre à prévenir la falsification ou l'usurpation d'identité.

Les utilisateurs commencent à comprendre la nécessité de protection des données numériques telles que les dossiers médicaux, fiscaux, financiers, des données personnelles, etc. À l'heure actuelle, où la plupart des services et des tâches peuvent être réalisés sans quitter la maison en utilisant des solutions numériques, la nécessité de sécuriser la transmission et le stockage des données croît.

Un autre problème associé à la sécurisation des données est l'influence de mécanismes de sécurisation sur le fonctionnement du système, le transfert ou l'échange de données. Les mécanismes de sécurisation de données (systèmes cryptographiques), qui ralentissent ou qui surchargent des systèmes sont très souvent désactivés et ne sont donc pas utiles.

Pour protéger effectivement les systèmes numériques, leurs concepteurs doivent prendre en compte non seulement l'efficacité ou robustesse des mécanismes de sécurité (contremesures), mais aussi leur impact sur la performance du système (vitesse, besoins en mémoire et en énergie). Malheureusement, avec l'augmentation de la sensibilisation à la sécurité des données et avec le développement de nouveaux mécanismes de sécurisation, le nombre de techniques de vol, d'interception et modification ou d'espionnage de données numériques augmente aussi.

La science qui étudie la sécurité de données est la cryptographie. Cryptographie est l'une des branches de la cryptologie. La deuxième branche de la cryptologie est la cryptanalyse. La cryptanalyse analyse des moyens de sécurisation proposés pour la cryptographie en vue d'évaluer leur robustesse (en particulier faisant des attaques).

Actuellement, pour sécuriser des données, les propriétés de problèmes mathématiques NP-difficiles et les variables spéciales qui rend la solution de ces problèmes très coûteux en temps et en ressources, sont utilisés. Avec l'augmentation de la puissance de calcul des systèmes numériques, certains problèmes mathématiques difficiles peuvent être résolu dans un délai raisonnable et pour un coût raisonnable (par exemple quelques années de calcul sur une machine parallèle). Pour augmenter la difficulté des problèmes, les

mathématiciens proposent d'augmenter la taille des nombres ou d'utiliser des nombres avec des propriétés spéciales telles que les nombres premiers.

Parce que les attaques mathématiques sont de plus en plus difficiles et plus coûteuses, de nouvelles méthodes (moins chères, plus faciles) pour violer des systèmes cryptographiques ont été proposées. Les cryptanalystes avaient commencé à s'intéresser aux dispositifs cryptographiques. Par analogie avec certaines techniques d'ouverture de serrures mécaniques (par exemple pour des coffres-forts), ils avaient tenté d'écouter le dispositif cryptographique au cours de son fonctionnement et d'analyser des données obtenues afin de déterminer s'il est possible de violer le système. Malheureusement les recherches de nouvelles techniques pour découvrir ainsi des secrets (données confidentielles, clés de chiffrement, etc.) étaient efficaces. Les cryptanalystes sont capables de décomposer un algorithme cryptographique ou d'en découvrir le secret par l'analyse de la consommation d'énergie du circuit ou de rayonnement électromagnétique.

Jusqu'à récemment, les fuites d'information de dispositifs numériques, telle que la consommation d'énergie instantanée ou du champ électromagnétique émis, ont été considérés comme un bruit négligeable, dont la réduction ou le contrôle n'a pas été correctement traité. À l'heure actuelle, les créateurs des systèmes cryptographiques sont conscients des risques de fuites d'information et ils cherchent à éviter ou contrôler ces éléments.

Les attaques qui utilisent la fuite d'information de dispositifs numériques pour violer des systèmes sont appelées attaques par canaux cachés (*side-channel attacks* - SCA). Ces attaques peuvent être divisées en catégories en fonction des fuites qu'elles exploitent. Il y a des attaques temporelles (*timing attacks*) qui analysent la durée des opérations, les attaques par analyse de la consommation d'énergie instantanée (*power analysis attacks*) et les attaques par analyse du rayonnement électromagnétique (*electromagnetic attacks*).

Les attaques par canaux cachés qui analysent la consommation d'énergie instantanée de dispositifs cryptographiques sont l'un des types les plus populaires d'attaques, dont beaucoup de variantes ont été développées. Elles

sont relativement faciles à réaliser, peu coûteuses et très efficaces. Développer des contre-mesures pour des systèmes cryptographiques contre les attaques physiques n'est pas une tâche facile. Concernant la grande complexité des algorithmes/systèmes cryptographiques, les dispositifs sont déjà assez complexes sans les contre-mesures ajoutées. L'ajout de dispositifs de protection comme des éléments tels que des générateurs de bruit peut rendre le dispositif encore plus complexe et lent, ce qui n'est pas acceptable. Additionnellement, il existe des techniques efficaces pour filtrer des bruits. Ceci rend souvent inefficace cette contre-mesure utilisée de façon autonome. Ce type de protection/contre-mesure peut être juste une contre-mesure supplémentaire. Des contre-mesures ajoutés/developpés doivent être soigneusement évaluées pour qu'elles n'aient pas un impact négatif sur l'efficacité du système cryptographique (occupation des ressources et la vitesse d'opération).

Les deux systèmes cryptographiques les plus populaires sont des systèmes de cryptographie symétrique (cryptographie à clé secrète) et systèmes de cryptographie asymétrique (cryptographie à clé publique). La sécurité de ces deux types de systèmes basés sur la sécurité de clés. La sécurité de clés est assurée par des problèmes mathématiques d'une très grande complexité (NP-difficiles) ne pouvant être résolus même avec une grande puissance de calcul dans un délai raisonnable.

Dans cette étude, nous envisageons des solutions dédiées aux techniques de cryptographie à clé publique (PKC). Dans PKC, la clé secrète (clé privée) peut être mathématiquement déterminée par la résolution d'un problème NP-difficile, tel que :

- le problème de factorisation des grands entiers (système RSA [1]) ;
- le problème du logarithme discret (DLP [2]) (système ElGamal)
- le problème du logarithme discret sur une courbes elliptique (ECDLP [3])

1. Rivest-Shamir-Adleman
2. discrete logarithm problem
3. elliptic curve discrete logarithm problem

(système ECC [4] - cryptographie sur les courbes elliptiques)

Dans ce travail, il a été décidé de se concentrer sur l'amélioration de l'efficacité et la sécurité des systèmes ECC. Principalement car il a été prouvé qu'ECC peut être bien plus efficace que RSA par exemple.

Les systèmes ECC sont considérés comme potentiellement plus efficaces en raison de la taille des clés qui sont utilisées pour assurer certain niveau de sécurité. Par exemple RSA utilisant des clés de 3072 bits fournit un niveau de sécurité équivalent à celle assurée par un système ECC utilisant des clés de seulement 256 bits. La comparaison des tailles de clés qui assurent la sécurité équivalente de differents systèmes cryptographiques est présentée dans la Table 1.

TABLE 1.: Tailles des clés de différentes systèmes cryptographiques assurant un certain niveau de sécurite [25]

sécurite (bit)	min. taille (bit) de clé (PKC)		
	DSA/DH	RSA	ECC
80	1024	1024	160
112	2048	2048	224
128	3072	3072	256
192	7680	7680	384
256	15360	15360	512

Les systèmes ECC utilisent des propriétés des courbes elliptiques définies sur des corps finis. Les principales opérations des protocoles ECC sont effectuées sur des points de telles courbes. Pour par exemple, pour pouvoir faire la somme de deux points de la courbe $P + Q$, il faut effectuer des opérations sur les coordonnées des points P et Q. Les coordonnées de ces points sont des éléments du corps finis sur laquelle la courbe est définie. Au vu de ces dépendances, nous pouvons conclure que les opérations sur les courbes elliptiques dépendent fortement sur des opérations sur les éléments des corps finis. Nous présumons que l'efficacité du système ECC dépend

4. elliptic curve cryptography

aussi fortement de l'efficacité des opérations arithmétiques dans les corps finis. Pour améliorer l'efficacité des systèmes ECC numériques, nous devrons proposer des opérateurs arithmétiques sur les corps finis rapides et avec des surfaces de circuit limitées pour des implantations en matériel. Deux types de corps finis qui sont utilisés en cryptographie sur les courbes elliptiques : les corps premiers $GF(p)$ et les extensions du corps fini binaire $GF(2^m)$. Dans ce travail, il a été décidé de considerer uniquement les extensions du corps fini binaire. Ces corps sont considérés comme les plus appropriées pour des solutions matérieles du fait que leurs éléments peuvent être représentées efficacement par des polynômes binaires, et non pas, comme dans le cas des corps premiers par des grands nombres. De cette façon, on évite les problèmes liés aux propagations de retenues.

Le premier objectif de la recherche était de développer en matériel des opérateurs arithmétiques sur $GF(2^m)$ performants (rapides et avec une surface de circuit limitée). La deuxième objectif est de rendre ces opérateurs plus robustes à certains attaques par canaux cachés de type d'analyse de consommation d'énergie.

L'opération la plus importante et la plus fréquemment attaquée dans les protocoles ECC est la multiplication scalaire d'un point de la courbe elliptique P par un très grand entier k, généralement secret (souvent il s'agit d'une clé privée) : $[k]P$. Cette opération est réalisée sous forme de chaîne d'opérations élémentaires sur les points de la courbe $(P+Q,\, 2P)$. Le plus simple des algorithmes pour la multiplication scalaire est un algorithme dit *double-and-add*, voir l'algorithme 1.

Dans cet algorithme, l'addition de points de la courbe (ADD) est strictement dépendante de la valeur du bit spécifique du scalaire secret k. Pour cette raison, quand il est possible de faire la distinction entre la consommation instantanée pendant des opérations d'addition (ADD) et de doublement (DBL), alors il est aussi possible d'une manière simple de découvrir le scalaire k (Figure 1).

Les opérations $P+Q$ et $2P$ sont effectuées sur les coordonnées des points

Algorithm 1 Algorithme double-and-add [9]

Input: $k = (k_{t-1}, ..., k_1, k_0)_2$, $P \in E(\mathbb{F}_q)$
Output: $[k]P$

1: $Q \leftarrow \infty$
2: **for** $i = 0$ to $t - 1$ **do**
3: **if** $k_i = 1$ **then**
4: $Q \leftarrow Q + P$ // ADD //
5: **end if**
6: $P \leftarrow 2P$ // DBL //
7: **end for**
8: **return** Q

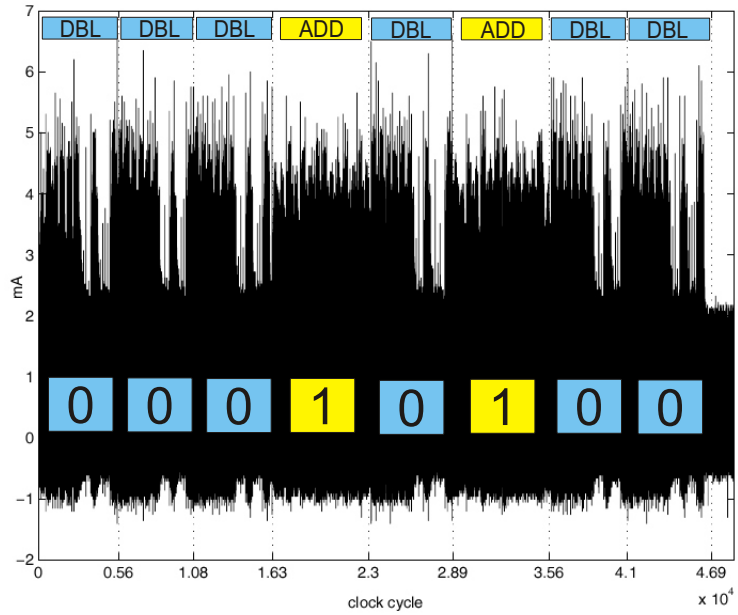

FIGURE 1.: L'exemple de SPA (analyse simple de consommation) d'algorithme 1

de courbes elliptiques. Les coordonnées des points des courbes elliptiques sont des éléments de corps fini, donc nous concluons que l'activité électrique (consommation d'énergie) pendant les opérations sur les éléments des corps

finis, pourraient donner des informations à un attaquant l'aidant ainsi à faire la distinction entre les opérations $2P$ et $P + Q$, et enfin l'aidant à decouvrir le clé secrète.

Notre tentative de sécuriser les opérateurs arithmétiques au niveau du corps fini est la première à notre connaissance. Dans la littérature, ne sont présentées que des techniques pour sécuriser les opérations au niveau de courbes elliptiques ou au niveau des protocoles ECC. Des contre-mesures existantes utilisées comme contre-mesures autonomes sont généralement suffisantes pour protéger les dispositifs contre les attaques par l'analyse simple de consommation (SPA), mais pour protéger ces dispositifs contre des attaques différentielles (DPA), il faut combiner plusieurs contre-mesures en plusieurs niveaux d'opérations. Les attaques SPA analysent une seule signature (trace) de consommation du dispositif, alors que les attaques DPA analysent un grand nombre de mesures (p. ex. quelques milliers) en utilisant un modèle fonctionnel du dispositif et des méthodes statistiques. On présume que la sécurisation de tous les niveaux des opérations de systèmes ECC devrait empêcher l'exécution fructueuse de telles attaques.

Pour cette raison, il a été décidé d'analyser et de réduire la fuite de l'information existant dans les opérateurs arithmétiques. Il est supposé que la réduction des fuites d'information des opérateurs arithmétiques peut améliorer la sécurité des systèmes ECC. Le troisième objectif, strictement connecté aux deux précédents, est de trouver un compromis entre l'efficacité et la sécurité des opérateurs. Généralement, plus on ajoute des contre-mesures (plus robuste est le dispositif), moins efficace est le dispositif.

En tenant compte de tous les objectifs présentés, nous pouvons formuler la thèse de nos recherches :
Il est possible de développer des opérateurs arithmétiques sur le corps fini $GF(2^m)$ *en matériel, efficaces et robustes contre les attaques aux canaux cachés qui analysent la consommation d'énergie.*

2. Recherches réalisés, problèmes rencontrés et les solutions développées

2.1. Opérateurs arithmétiques sur $GF(2^m)$ en matériel.

Il existe deux opérations principales définies sur $GF(2^m)$: addition et multiplication. Toutes les autres opérations division, inversion, etc.) peuvent être vues comme une chaîne d'additions et de multiplications. Avant de réaliser le premier objectif (développement d'opérateurs arithmétiques sur $GF(2^m)$ en matériel), il était nécessaire de trouver des paramètres/caractéristiques des opérateurs qui nous permettent de développer des opérateurs efficaces.

Des paramètres vitaux pour les opérateurs arithmétiques sont : la base (représentation de nombres), le polynôme irréductible $f(x)$ (générateur du corps) et la taille de corps utilisé m (taille des nombres sur lesquels les opérateurs vont opérer).

Le type de base utilisé dépend des applications des opérateurs. Les bases les plus populaires en cryptographie sont : la base polynomiale, la base normale ou quasi-normale (Gaussienne GNB, optimale ONB) et la base duale. Après avoir effectué une vaste analyse détaillée des propriétés et applications de différentes bases, nous avons décidé d'utiliser la base polynômiale de forme : $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$. En premier lieu, la base duale a été rejettée. Généralement, cette base est utilisée pour des petits corps finis ($m = 8, 16, 32$) et dans nos recherches nous considérons des corps avec $m \cong= 150 - 600$. Les bases polynomiale et normale ont des caractéristiques qui pourront rendre nos opérateurs très efficaces. Finalement en considérant des opinions trouvés dans la littérature (que la base polynomiale permet d'obtenir des solutions matérielles plus efficaces), nous avons décide d'utiliser la base polynomiale. Néanmoins nous envisageons d'analyser l'utilisation des bases normales dans l'avenir.

Le deuxième paramètre important est la taille m du corps fini et du type de polynôme irreducible $f(x)$ utilisé. En vue d'applications ECC, il faut utiliser des tailles m et des polynômes $f(x)$ définis dans des standards

cryptographiques comme ceux du NIST ou SEGC [7].

Toutes les solutions proposées sont dédiées aux circuits reconfigurables, en particulier aux circuits FPGAs. Nous avons utilisé des circuits Xilinx : Spartan-3E XC3S1200E [29] et Virtex-6 LX240T [2]. Pour des évaluations de sécurité une carte SASEBO-G [22] avec un FPGA Virtex-II Pro XC2VP7 [28] a été utilisée.

Opérateur d'addition. L'addition sur $GF(2^m)$ est une simple opération. Cette opération peut être vue comme un simple `XOR`. D'un autre côté, même une simple opération mathématique effectuée sur des éléments très grands peut causer des problèmes de synchronisation (qui peut ralentir d'autres fonctions) et peut occuper beaucoup de ressources. Afin de vérifier l'impact réel de l'opération d'addition sur d'autres composants de systèmes ECC, nous avons préparé plusieurs types d'opérateurs d'addition.

Le processeur ECC, dans lequel nos opérateurs seront integrés, transfère des données en mots (16, 32 bits). Pour cette raison, les solutions suivantes ont été proposées :

1 : Addition des mots d'entrée, les vecteurs a, b, et stockage des résultats partiels dans le registre c (solution 1).

2 : Addition des mots d'entrée, les vecteurs a, b, et stockage des résultats partiels dans la mémoire (solution 2).

3 : Attente des vecteurs a, b complets et addition des vecteurs reçus (solution 3).

Dans la Table 2, les résultats obtenus sont présentés. Les paramètres de la solution 3 ne sont pas présentés en raison du fait que la solution est très simple, on peut dire qu'il s'agit d'une translation des entrées vers les sorties du circuit.

Les résultats obtenus montrent que l'addition est une opération simple, ayant un effet négligeable sur les autres composants du système. Les solutions proposées sont petites (de 20 à 30 LUT) et rapides. Avec l'augmentation de la taille des vecteurs l'utilisation de ressources est plus un peu plus

TABLE 2.: Opérateurs d'addition (Virtex-6)

taille du corps m	solution 1		solution 2	
	[LUT]	[MHz]	[LUT]	[MHz]
163	21	771	26	562
233	21	771	26	562
283	22	767	28	560
409	22	767	28	560
571	24	578	31	558

importante et la vitesse baisse légèrement.

L'addition des éléments du corps fini dans les systèmes ECC est généralement effectuée en parallèle avec d'autres opérations en raison du faible degré de sa complexité.

Opérateur de multiplication sur $GF(2^m)$. La multiplication dans un grand corps fini est une opération complexe. Elle est considérée comme une opération modulaire, afin d'obtenir le produit de deux éléments du corps, il faut d'abord les multiplier, et ensuite réduire le produit modulo le polynôme irréductible $f(x)$. De cette façon, le résultat de la multiplication de deux éléments du corps $c(x) = a(x)b(x) \bmod f(x)$ est aussi un élément du même corps. On divise les algorithmes de multiplication en deux groupes : les algorithmes à deux étapes et les algorithmes entrelacés. Plusieurs d'algorithmes existent et leurs modifications ont été analysées en regardant/cherchant leurs propriétés (traitement de données, représentation des données, manières de paralléliser, etc.) qui pourront être utiles pour développer des multiplieurs matériels très efficaces/performants. Cette section résume les principaux résultats obtenus et conclusions tirées de l'analyse des algorithmes décrits dans la littérature disponible.

Les solutions proposées ne seront pas totalement nouvelles car elles se-

ront basées sur des théories mathématiques bien connues. Le but de cette recherche est de modifier ou combiner certaines des propriétés existantes dans le but de développer des algorithmes pour les opérateurs matériels plus performants.

Algorithmes à deux étapes - des résultats les plus intéressants

Au cours de notre étude, nous avons examiné séparément des méthodes de multiplication et de réduction.

Premier étape : multiplication. La méthode de la multiplication de polynômes la plus connue est la méthode "shift-and-add". Etant donnés deux polynômes $a(x)$ et $b(x)$ de degré $(m-1)$, le produit $d(x)$ de degré $(2m-2)$ est obtenu par multiplication du polynôme $a(x)$ par chaque coefficient du polynôme $b(x)$. Plus simplement, on décale le polynôme $a(x)$ et on additionne le nouveau polynôme si le coefficient correspondant de $b(x)$ est `1`. Cette méthode est très simple et permet de concevoir des circuits avec des structures régulières. Cependant, utilisé dans une forme inchangée pour multiplier des très grands vecteurs, elle peut ne pas être très efficace, surtout en termes d'utilisation de ressources matérielles. La plupart des algorithmes de multiplication de polynômes sont généralement des variantes de cet algorithme. Les différences entre les différentes versions s'appuient sur des méthodes de représentation des nombres, la façon de traiter les données, etc. Dans le cas de la multiplication de grands nombres, on utilise par exemple une méthode de partitionnement des données de type "divide-and-conquer", des représentations matricielles des arguments, ou bien des moyens spécifiques de traitement de données. Les optimisations les plus intéressantes semblent être : une méthode basée sur le principe " divide-and-conquer", l'optimisation de Karatsuba-Ofman et la méthode matrice-vecteur.

Le principe de la méthode "divide-and-conquer" est de diviser un gros problème en petits sous-problèmes. Dans notre cas, on divise le problème de multiplication de grands nombres en un ensemble de plus petites multiplications, et ainsi de partitioner les vecteurs d'entrée en plus petits vecteurs. Les produits partiels sont ensuite combinés pour obtenir le résultat final. La

variante la plus populaire et la plus utilisé est l'optimisation de Karatsuba-Ofman [10]. Elle réduit le nombre des opérations nécessaires à effectuer afin d'obtenir le résultat final.

Dans la méthode matrice-vecteur [4], le polynôme $a(x)$ est représenté par une matrice A de taille $(2m-1)\times m$, dans laquelle chaque colonne représente un nouveau décalage vers la gauche du vecteur $a(x)$ ($a << 2$). L'élément $b(x)$ est représenté par un vecteur de taille m et le produit $d(x)$ et aussi un vecteur mais de taille $(2m-1)$.

Les représentations des nombres influent fortement sur les architectures des solutions matérielles. Certaines permettent d'accélérer le fonctionnement du système, mais augmentent la quantité des ressources matérielles occupées, d'autre diminuent de manière significative le nombre des ressources utilisées, mais diminuent aussi la vitesse de la solution. Lors de la conception d'architectures matérielles, il faut toujours trouver un compromis entre la taille de la solution et la vitesse de fonctionnement. Afin de concevoir des opérateurs de multiplication efficaces, il était nécessaire d'étudier un grand nombre de solutions existantes. Malheureusement, la majorité des optimisations d'algorithmes présentés dans la littérature sont uniquement des considérations théoriques. Souvent, des optimisations théoriques présentées ont donné des résultats plus mauvais ou similaires à des solutions originales plus simples. Nous avons réalisé et analysé un grand nombre de solutions afin de trouver les propriétés des algorithmes qui peuvent être bénéfiques pour le matériel. Une analyse détaillée des solutions différentes pour les multiplieurs a été présentée lors de conférences nationales et internationales [17, 16, 18].

Pour faciliter la comparaison des solutions présentés, nous avons introduit un coefficient AT (produit temps surface comme coefficient de performance), défini comme suit : AT = (ressources$\times$temps d'opération). Les résultats indiquent que les meilleures solutions pour les algorithmes à deux étapes peuvent être obtenues en utilisant la méthode matrice-vecteur pour la représentation des nombres et l'optimisation de Karatsuba-Ofman (op-

timisation “divide-and-conquer”). Les résultats obtenus pour les opérateurs de multiplication pour les corps de tailles 233, 283, 409, 571 sont présentés dans la Table 3.

TABLE 3.: opérateurs de multiplication 233, 283, 409 et 571 bits (Virtex-6)

Multiplieur	Taille [LUT]	Max.f(#cycles) [MHz]	AT
233-bit opérateur composé de trois sous-multiplieurs 117 bits	2625	520 MHz (234)	1181.3
283-bit opérateur composé de trois sous-multiplieurs 142 bits	3381	535 MHz (284)	1794.8
409-bit opérateur composé de trois sous-multiplieur 206 bits	4834	535 MHz (412)	3722.6
571-bit opérateur compose de trois sous-multiplieurs 286 bits	7095	522 MHz (572)	7774.6

Deuxième étape : réduction modulo $f(x)$. Généralement, il existe deux méthodes de réduction : classique et matrice R. La première méthode, la méthode classique, est la méthode itérative. La seconde deuxième utilise une matrice R de réduction [4]. Plusieurs optimisations ont été proposées pour les deux méthodes, la plupart optimisent le nombre d'opérations nécessaires pour calculer avec un polynôme irréductible spécifique $f(x)$. Il existe des types particuliers de générateurs de corps $f(x)$ comme les trinômes, pentanômes ou par exemple des polynômes de type ESP [5], qui permettent une simplification significative de l'algorithme de réduction. Dans la thèse, nous proposons un moyen d'optimiser l'algorithme de réduction (en vue de polynôme irréductible) qui réduit le nombre des opérations nécessaire à effectuer.

La deuxième méthode de réduction utilise une matrice specifique R [4]. Cette méthode peut considérablement accélérer la réduction modulo $f(x)$. La matrice de réduction R est définie par le polynôme irreducible $f(x)$. Il est généralement admis que la méthode de réduction utilisant la matrice R

5. equally spaced polynomials

est très efficace. Au cours nos recherches, la méthode était optimisée pour $f(x)$.

Afin de comparer des méthodes de réduction, nous avons implémenté en matériel la version classique, optimisée et la méthode utilisant la matrice R. Les résultats sont présentés dans la Table 4. L'unité basée sur

TABLE 4.: Les résultats d'implémentation des algorithmes de réduction ($m = 233$) (Virtex-6)

Algorithme	Ressources [LUT]	Max. f(#cycles) [MHz]	AT
classique	3528	209 MHz (600)	10128.2
classique, optimisé	1165	571 MHz (8)	16.3
matrice R (séquentiel)	466	1264* MHz (2)	0.74
matrice R (combinatoire)	233	1.13 ns	0.26

* Les résultats présentés dans la table sont les résultats donnés par l'environnement Xilinx ISE, dans ce cas, les résultats nous disent que les solutions après l'intégration avec d'autres circuits n'auront aucun effet sur la vitesse du système.

la méthode classique non-optimisée a besoin de beaucoup de cycles pour effectuer l'opération, et occupe beaucoup de ressources. Dans le cas de l'algorithme optimisé, le nombre d'opérations nécessaires pour effectuer une reduction dépend de la forme du polynôme irréductible. Pour $m = 409$ et $m = 233$, le nombre d'opérations nécessaires a effectuer est le même. Pour $m = 163,\ 283,\ 571$, le nombre d'opérations nécessaires a effectuer croît jusqu'à 14. En résumé, on peut dire que si le polynôme irréductible est défini, il est possible d'optimiser la méthode de réduction.

La solution la plus efficace semble être la solution utilisant la matrice de réduction R. Pour les polynomiaux irréductibles définis dans les standards/normes ECC, la matrice de réduction R contient beaucoup de zé-

ros. La solution la plus efficace semble être la solution utilisant la matrice de réduction R. Pour cette raison, une opération compliquée et qui prend beaucoup de temps peut être remplacée par un chaîne d'opérations `XOR`. Seulement les éléments non-nuls du polynôme irréductible nécessittent des opérations `XOR`.

Résumé de l'analyse des algorithmes à deux étapes. L'analyse a abouti à la conception d'opérateurs de multiplication sur $GF(2^m)$. En analysant les résultats obtenus, il a été décidé que les meilleures composants pour créer un opérateur performant sont :

- **multiplieur** : solution utilisant la méthode matrice-vecteur, composé de trois sous-multiplieurs utilisant l'optimisation Kartasuba-Ofman (partition des vecteurs en deux $(m/2 \pm 1)$) ;
- **circuit de réduction** : solution utilisant l'algorithme classique optimisé avec le polynôme irréducible ou la solution utilisant la matrice de reduction R optimisée pour le polynôme irréducible.

Les résultats des implémentations des opérateurs de multiplication sur $GF(2^{233})$ sont présentés dans la Table 5.

TABLE 5.: Opérateurs de multiplication $GF(2^{233})$ (Virtex-6)

Multiplieur m=233	Ressources [LUT]	Max. f [MHz]	# cycles d'horloge	AT
opérateur avec réduction classique	3638	302	264	3.18×10^3
opérateur utilisant matrice R	2862	302	238	2.25×10^3

Algorithmes entrelacés (interleaved) - des résultats les plus intéressants

Dans ce type d'algorithmes, l'opération de multiplication est effectuée en alternance avec la réduction. Les algorithmes les plus intéressants de ce groupe semblent être : l'algorithme utilisant la matrice de Mastrovito et l'algorithme de Montgomery.

L'algorithme utilisant la matrice de Mastrovito. L'algorithme utilisant la matrice de Mastrovito est une extension de la méthode matrice-vecteur. Dans cet algorithme, à la place de deux étapes de multiplication et réduction on calcule : $c = Mb$, où M est *la matrice de Mastrovito.*

La matrice de Mastrovito est composée de la matrice A (représentant $a(x)$) et de la matrice de réduction R [4]. À la suite de la recherche, une optimisation efficace de l'algorithme a été développée. Nous avons proposé d'au lieu d'opérer sur et de stocker la matrice entière M, de la partitionner en sous-matrices (sous-blocs) de la taille 16×16 bits. Un exemple de partition de matrice M pour le corps $m = 233$ est présenté en Figure 2 Les sous-matrices et la partie corespondante du vecteur b sont multipliées

M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	M1	Mc
1	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	M1	Mc
2	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	M1	Mc
3	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M0	M1	Mc
4	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
5	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	M4	Mc
6	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	M4	Mc
7	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	M4	Mc
8	M0	M0	M0	M2	M2	M2	M2	M2	M4	M4	M4	M4	M4	M4	Mc
9	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	M3	Mc
10	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	M3	Mc
11	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	M3	Mc
12	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	M3	Mc
13	M0	M0	M0	M0	M0	M0	M0	M0	M2	M2	M2	M2	M2	M3	Mc
14	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR	MR

FIGURE 2.: Illustration de partition de matrice de Mastrovito pour $m = 233$

par des unités spécifiques de sous-multiplieurs. Chaque type de sous-matrice (même index en Figure 2) nécessite/demande un type spécifique de sous-multiplieur.

En plus, les coefficients des sous-matrices sont calculés pendant la multiplication. Ainsi on évite de stocker une grande quantité de données. Le processus de multiplication de toutes les sous-matrices est contrôlé par automate fini (FSM). Pour trouver la meilleure solution matérielle basée sur la conception de Mastrovito, nous avons implementé et analysé des nombreuses variantes de ce type de solution. Nous avons modifié le nombres des états d'automate fini, le nombre de sous-multiplieurs de différents types utilisés, etc. En raison de l'énorme quantité de variantes de cette solution, nous n'avons pas testé toutes les modifications possibles. Pour conclure, l'analyse a révélé que la solution la plus efficace utilise l'automate fini avec plus d'états, mais en utilisant une seule sous-unité pour chaque type de sous-multiplieur (il est par exemple aussi possible d'utiliser un automate fini avec moins d'états et plusieurs unités sous-multiplieurs de même type).

TABLE 6.: Solutions utilisant la matrice de Mastrovito

Virtex-6 XC6VLX240T	$(A^L b + A^H R)b$	Mb
ressources(A)[LUT]	5014	3760
max.f	297 MHz	276 MHz
durée des opérations (T)[nb. cycles]	65	75
AT	1097	1021

L'algorithme de Montgomery. La deuxième solution, aussi très populaire, pour les algorithmes entrelacés est l'algorithme de Montgomery [14]. La méthode suppose que pendant plusieurs opérations, pendant la chaîne des opérations de multiplication, on opère sur des nombres en représentation de Montgomery. Dans la représentation de Montgomery, pour effectuer la multiplication modulaire $c(x) = a(x)b(x) \bmod f(x)$, nous effectuons l'opération $c(x) = a(x)b(x)r^{-1}(x) \bmod f(x)$, où $r(x)$ est un paramètre spécifique. Si $r(x)$ est convenablement choisi, l'opération de multiplication devient simple. Pour recevoir le "vrai" résultat de multiplication sur corps fini,

il faut effectuer la réduction de Montgomery (retrouver la représentation classique à partir de la représentation de Montgomery). Pour effectuer une seule multiplication, en utilisant cet algorithme il faut utiliser l'algorithme deux fois (deux exécutions de cet algorithme donne le résultat de multiplication modulaire sur le corps fini). L'algorithme complet, pour effectuer une seule multiplication dans le corps $GF(2^m)$, est présenté ci-dessous, voir l'algorithme 2.

Algorithm 2 Multiplication modulaire - méthode de Montgomery [11]

Input: $a(x), b(x), r(x), f(x), f'(x), r^2(x) \bmod f(x)$
Output: $c(x) = a(x)b(x) \bmod f(x)$
1: $t(x) = a(x)b(x)$
2: $u(x) = t(x)f'(x) \bmod r(x)$ // $MontMult(a, b)$ //
3: $d(x) = [t(x)$ `XOR` $u(x)f(x)]/r(x)$
4: $t(x) = d(x)(r^2(x) \bmod f(x))$
5: $u(x) = t(x)f'(x) \bmod r(x)$ // $MontMult(d, r^2 \bmod f)$ //
6: $c(x) = [t(x)$ `XOR` $u(x)f(x)]/r(x)$
7: **return** c

L'algorithme utilise trois coefficients additionnels : $r(x)$, $r^2(x) \bmod f(x)$, $f'(x)$. L'élément $r(x)$ selon [11] pour $GF(2^m)$ est un simple monôme x^m. Si le polynôme $f(x)$ est défini auparavant, des coefficients constants $f'(x)$ et $r^2(x) \bmod f(x)$ peuvent être calculés.

La Table 7 présente des résultats obtenus pour les implémentations de la méthode Mastrovito. L'opération le plus complexe de cet algorithme, la multiplication de deux grands vecteurs a, b, s'effectue en étape deux (ligne deux de l'algorithme 2). Par conséquent, la multiplication de deux grands vecteurs a, b a un impact significatif sur la vitesse et l'occupation de ressources.

2.2. Résume et conclusions

La comparaison des solutions proposées avec des solutions existantes n'est pas facile. La documentation disponible ne contient pas toujours toutes les

TABLE 7.: L'algorithme de Montgomery - solutions (Virtex-6)

ressources LUT	max.f MHz	# cycles	multiplieur	AT
3197	338	270	233-bit operateur compose de 233x16-bit sous-multiplieurs ressources : 2308 LUT, max.f : 323MHz	2554
3730	302	244	233-bit operateur compose de trois 117-bit sous-multiplieurs ressources : 2625 LUT, max.f : 302MHz	3014

données nécessaires. Les solutions ne sont pas toujours complètement décrites, il manque la durée des opérations, la vitesse, ou il y a des unités differentes. Dans de nombreux articlesm des performances de dispositifs sont evaluées seulement théoriquement. Parce que les résultats de l'implementation dépendent non seulement de l'algorithme utilisé, mais aussi sur la manière dont il est décrit dans le langage HDL, notre implémentation des algorithmes trouvés dans la litterature peut donner des résultats différents de ceux obtenus par leurs auteurs. La Table 8 présente les paramètres des solutions proposées à la suite de nos recherches

TABLE 8.: Multiplieurs sur $GF(2^{233})$ proposées

Algorithme	Ressources [LUT]	f [MHz]	cycles d'horloge	AT
Classique 1	3638	302	264	3.18
Classique 2	2862	302	238	2.25
Mastrovito	3760	297	75	0.95
Montgomery (complet)	3197	338	270	2.55

La Table 9 présente les paramètres des solutions trouvées dans la littérature disponible. En comparant le contenu de ces deux tables, nous pouvons conclure que les solutions proposées à la suite de nos recherches sont assez rapides et nécessitent des quantites raisonnables de ressources.

TABLE 9.: Solutions trouvées dans la littérature

	m	Circuit	ressources	Max.f /delai	T
[3]	256	Virtex II 2000-6	5267 LUT	44.91 MHz	5.75 us
	1033				23.07 us
[26]	1024	XCV2000E-6	4355 CLB	100.4 MHz	-
		XC40150XV-7	8339 CLB	44.4 MHz	-
		XC4VFX100-10	2793 CLB	150.5 MHz	-
[5]	233	XC2V-6000-4	415 slices	-	2.42 us
[15]	233	Stratix EP1S40F780C5	3728 LE	4.04 ns	12 cycles
	283		3396 LE	3.66 ns	20 cycles
[27] by [15]	233	Stratix EP1S40F780C5	3353 LE	6.91 ns	16 clock
	283		3118 LE	6.95 ns	20 clock
[8]	233	XC2V-6000 FF1517-4	37296 LUT	77 MHz	-
			11746 LUT	90.33 MHz	-
			36857 LUT	62.85 MHz	-
			45435 LUT	93.20 MHz	-
[23]	191	XCV2600E	8721 CLB	-	82.4 us
[1]	88	Altera EP2S60	6644 ALUTS	-	
[6]	163	Virtex XCVL330	201,989 LUTs 214,703 DFFs	241 MHz	-
			1471 LUTs 982 DFFs	241 MHz	- -
[12]	283	Virtex 4 XC4VFX140	1781 CLB 2156 FF 3367 LUT	246.670 MHZ	-
	1132		25,955 CLB 32,578 FF 48,591 LUT	248.447 MHz	-

3. Sécurisation de opérateurs arithmetiques matérielles sur $GF(2^m)$

Le deuxième objectif de la thèse est de protéger des opérateurs arithmétiques contre des attaques par analyse de la consommation d'énergie. Les résultats présentés ci-dessous ont été présentés lors de la conférence WAIFI 2012 [19]. Les attaques par analyse de la consommation considère deux types de fuite d'information :

- *la fuite résultant du nombres de transitions* - nombre de bits qui changent leurs états dans une unité de temps,
- *la fuite résultant du changement de la distance Hamming HW* - nombre de bits avec état `1`, traités dans une unité de temps,

Parce que la consommation instantanée des circuits VLSI est liée au nombre de transitions qui se produisent dans une unité de temps, nous avons considéré la fuite résultant du nombre de transitions utiles qui se produisent pendant le fonctionnement de système (*Hamming distance* = $HW(t+1) - HW(t)$). Etant des transitions utiles, nous considérons le nombre de transitions des bits et signaux pendant les opérations arithmétiques. Nous nous focalisons sur des changements qui se produisent entre deux cycles d'horloge. Les transitions utiles étant définies, nous allons l'appeller *l'activité utile* du système. Pour identifier la source des fuites d'information qui existent dans les unités développées, nous avons d'abord proposé des moyens d'évaluation, puis nous avons effectué une analyse approfondie.

Dans les attaques par analyse de la consommation, l'attaquant essaie de trouver les corrélations qui existent entre la consommation instantanée du dispositif cryptographique et les données sur lesquelles la dispositif exerce ses opérations [13].

Afin d'évaluer le niveau de sécurité des opérateurs (évaluation de l'activité utile) la méthode suivante a été proposée. Il était proposé de surveiller chaque registre et signal en utilisant un *moniteur d'activité* [24].

Un moniteur d'activité compte le nombre des transitions des bits de la manière suivante : la bascule `D` mémorise l'état précédent du signal et la porte `XOR` compare l'état actuel du signal avec l'état mémorisé. La porte `XOR` donne en sortie `1` si il y a eu une transition. Le nombre des transitions est compté par un compteur k bits. La taille de compteur k dépend du nombre des signaux à regarder.

Compte tenu la spécificité des circuits VLSI, il s'avère que la méthode d'évaluation de l'activité du système avec des moniteurs d'activité donne des résultats très similaires à ceux obtenus en faisant une mesure de courant en utilisant une sonde et une oscilloscope [24]. Après avoir examiné les sources existantes de fuites d'information et des façons de les réduire, nous présentons une comparaison entre les données obtenues en utilisant la méthode des moniteurs et les données obtenues par la mesure de courant instantané consommé par le dispositif.

À la suite de nos recherches, nous avons proposés trois types des opérateurs arithmétiques sur $GF(2^m)$: classique, Mastrovito et Montgomery. La sécurite de ces opérateurs a été évaluée et les modifications pour augmenter la sécurite ont été proposées.

3.1. Vérification du niveau de sécurité, l'identification du problème, les modifications proposées

Nous avons effectué un grande nombre de tests en utilisant beaucoup de d'ensembles différents de données aléatoires avec une distribution uniforme de zéros et de uns. Les résultats présentés ici sont des résultats typiques/représentatifs. Il est noté que la correlation entre les courbes des activité obtenues et les données est petit. Malhereusement il était impossible de moyenner des résultats en raison du fait que cela pourrait fausser les résultats, par exemple par l'aplatissement ou l'unification de la courbe d'activité. La thèse présente l'analyse pour $m = 233$, pour les autres tailles de corps définis dans les normes, les résultats sont similaires.

Solution avec l'algorithme classique Nous avons proposé deux solutions classiques : une utilsant la réduction classique optimisée et la deuxième utilisant la matrice de réduction R.

Après avoir analysé des échantillons de l'activité pour les deux types de solutions, il a été constaté que la réduction n'a pas d'impact significatif sur la forme de la courbe d'activité. Les solutions pour les deux courbes sont presque identiques (pour les mêmes données de test). Ainsi, il a été décidé d'analyser la solution en utilisant des méthodes de réduction classique optimisée.

La figure 3 (figure du haut) montre l'activité de l'unité non sécurisée pour une séquence de multiplications effectuées sur des arguments aléatoires. Il est facile d'observer une source potentielle des informations utiles pour l'attaquant, une caractéristique qui peut faciliter la détermination des limites temporelles des opérations individuelles de multiplication, ou la détermination du nombre d'opérations de multiplication qui ont été réalisées.

Au début de chaque opération, il y a un accroissement soudain de l'activité, ce qui permet de déterminer le début d'opération de multiplication. Cet accroissement soudain de l'activité se produit par l'initialisation/réinitialisation du dispositif, par le manière dont les valeurs des arguments sur lesquels opère le dispositif sont rechargés.

Pour éviter un accroissement soudain de l'activité provoqué par l'initialisation, le contrôleur et la manière de réinitialiser l'operateur ont été modifiées. Auparavant, la réinitialisation et le rechargement des valeurs de tous les registres utilisés étaient faites au début de multiplication. Maintenant des registres remettent à zéro leur contenu immédiatement avant l'utilisation. Comme cela, le processus d'initialisation est réparti sur plusieurs de cycles d'horloge ce qui évite des changements des valeurs de tous les registres dans une unité de temps. La Figure 3, la partie inférieure, presente l'activité des operateurs avec les modifications proposées. Lors de nos recherches, nous avons developé plusieurs maniéres différentes pour l'initialisation. Dans le document, nous presentons l'activité pour la meilleure modification obtenue.

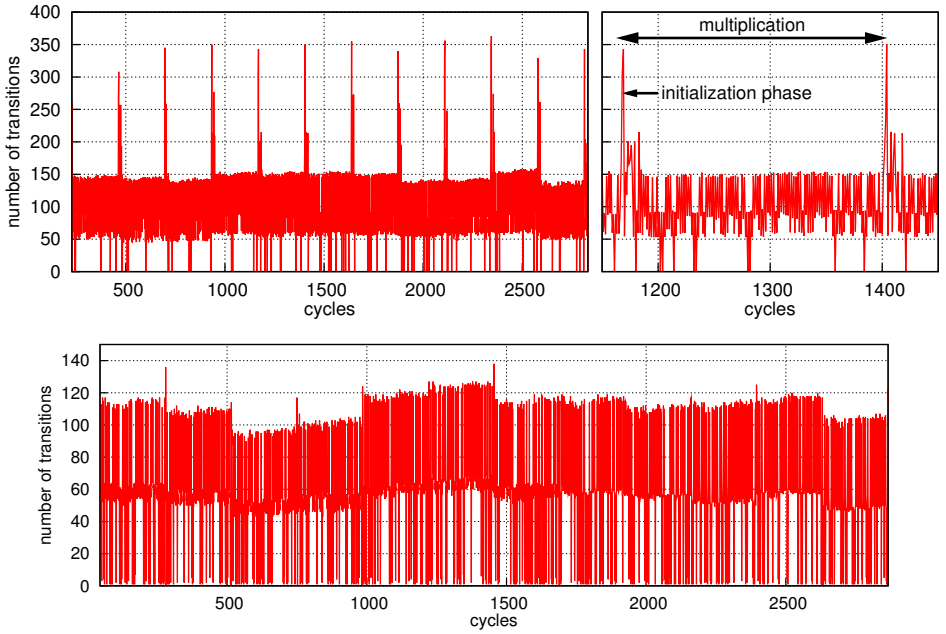

FIGURE 3.: L'activité utile de la solution classique

Les modifications effectuées ont permis de repérer d'autres manières d'optimiser des operateurs et d'apporter des nouvelles améliorations. Comme nous pouvons l'observer, le nombre de transitions a diminué, maintenant elle varie de 100 à 120 transitions, dans la version non modifiée elle a varie de 100 à 150 transitions. Les optimisations qui menent à une réduction du nombre de transitions réduisent légèrement la consommation d'énergie. L'analyse de la forme de la courbe d'activité obtenue pour la version sécurisée de l'opérateur montre un manque de croissance d'activité caractéristique. De plus, dans un intervalle entre 1500 et 2400 cycles, il est difficile de déterminer les bords temporels des opérations consécutives, il est difficile de déterminer le nombre de multiplications faites.

Solution basée sur l'algorithme de Montgomery L'activité de solution non protégée basée sur l'algorithme de Montgomeryest presenté dans la

Figure 4 (figure du haut). Pour comparer de façon adéquate la solution basée

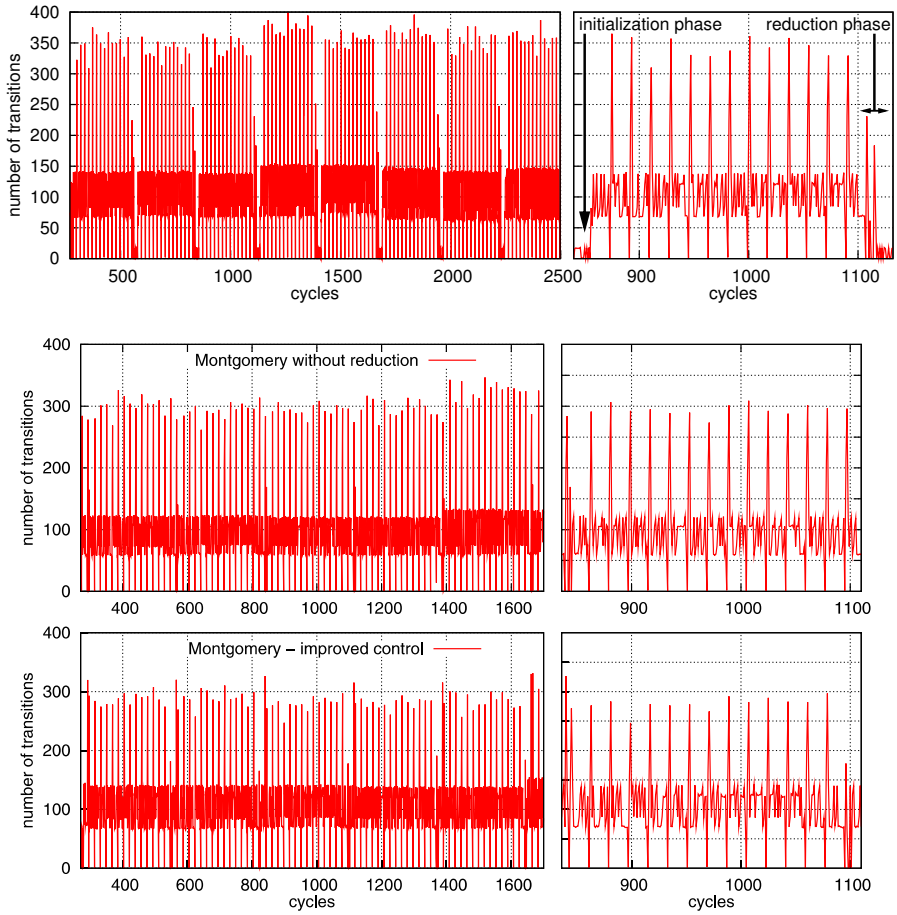

FIGURE 4.: L'activite utile de solution basé sur l'algorithme de Montgomery.

sur l'algorithme de Montgomery avec d'autres solutions, nous montrons ici des résultats obtenus pour le multiplieur complet. Chaque multiplication se termine par la conversion depuis la représentation de Montgomery.

Dans le cas de cette solution, on peut observer une diminution de l'activité à la fin de l'opération de multiplication. Il a été constaté que cette

baisse est due à l'opération de conversion de la représentation et due au contrôleur du multiplieur, qui arrête le multiplieur pour recharger de nouvelles valeurs des arguments. Dans la partie inférieure de la Figure 4, il est presenté l'activité du multiplieur complet avec un contrôleur amélioré (improved control), et l'activité du multiplieur avec un contrôleur amélioré et sans la conversion de la représentation Montgomery (without reduction). Si on néglige la conversion de la représentation de Montgomery, on peut observer que l'activité de solution est assez uniforme. L'activité ne contient pas les baisses drastiques ou des accroissements de l'activité qui pourraient aider à localiser des opérations dans le temps.

La solution utilisant la matrice de Mastrovito La Figure 5 (en haut) montre l'activité du multiplieur qui utilise les propriétés de la matrice Mastrovito.

L'activité du multiplieur a une forme très particulière de "dents de scie". Cette forme très particulière peut fournir à un attaquant beaucoup d'informations utiles. Il était nécessaire de modifier l'architecture de l'opérateur pour réduire ces variations spécifiques d'activité. Pour l'architecture de l'opérateur conçu (division sur les sous-matrices) nous avons proposé deux types des modifications : uniformisation (aplatissement) d'activité et randomisation d'activité (chaque multiplication aura une activité relativement différente). La première modification devra rendre plus difficile de localiser la début et la fin des opérations, et la deuxième devra rendre plus difficile la détermination de type d'opération effectué.

Il est évident que la forme de l'activité se produit en raison d'un nombre inégal de bits qui changent d'état dans une unité de temps. Le but de l'uniformisation est d'égaliser le nombre de bits qui changent d'état dans une unité de temps. Le but de randomisation et de randomiser le nombre de bits qui changent d'état dans une unité de temps. Nous avons proposé plusieurs modifications qui mènent à une uniformisation ou à une randomisation.

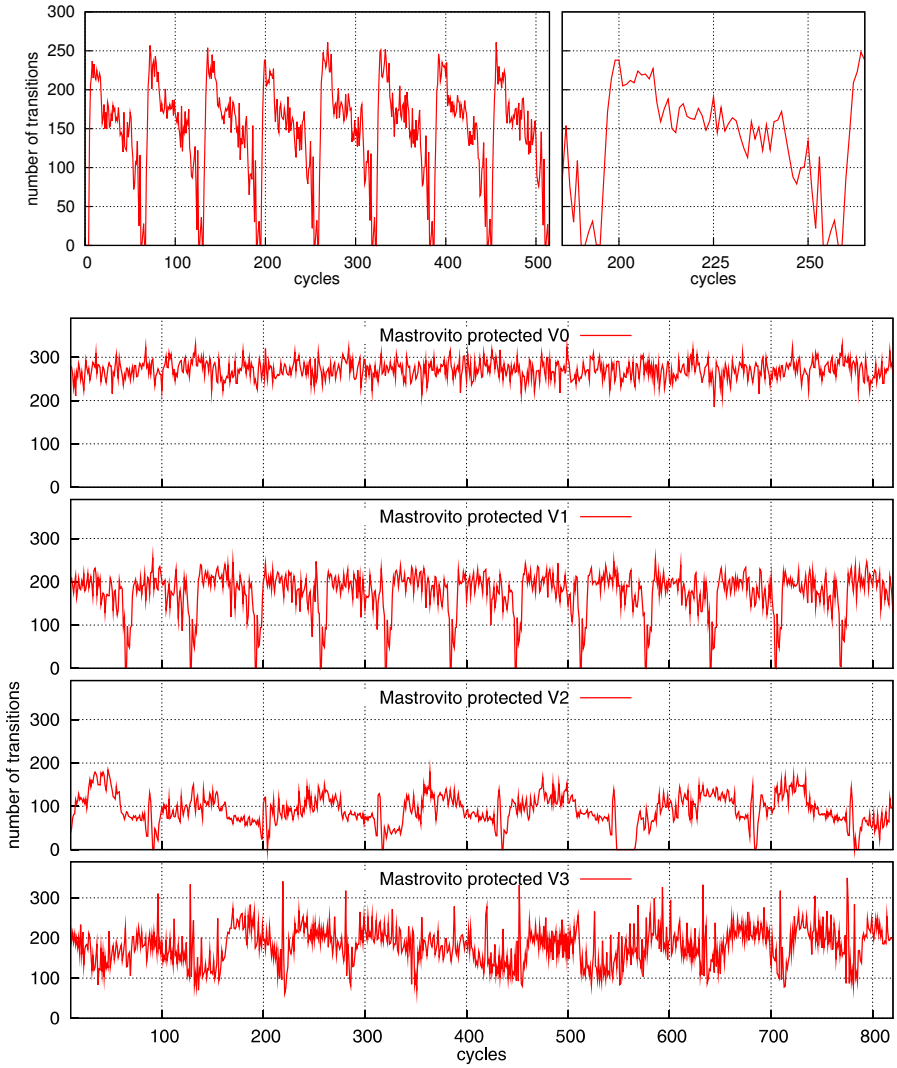

FIGURE 5.: L'activite des solutions basées sur la conception de Mastrovito (4 versions).

Dans ce résumé, nous présentons des résultats les plus intéressants.

Le meilleur résultat obtenu pour l'uniformisation, sans modifier le nombre

des états de la machine d'états (FSM) est présenté dans la figure 5, avec la courbe V1. Après avoir modifié le FSM, le nouveau moyen d'uniformiser l'activité est présenté par la courbe V0. À la suite des modifications réalisées pour randomiser l'activité, nous avons obtenu des courbes V2 et V3. De plus, la randomisation fait varier la durée de multiplication. Pour les modifications représentés par la courbe V2, le nombre de cycles d'horloge pour réaliser une multiplication varie de 98 à 126. Dans le cas de la courbe V3, le nombre cycles varie de 64 à 108.

Le but des modifications effectuées était de masquer des formes caractéristiques des opérations de multiplication. Pour vérifier le niveau de sécurité qui résulte de ces modifications, nous avons effectué quelques variantes populaires de cryptanalyse de consommation. Par exemple, nous avons analysé la corrélation entre la forme de la courbe de l'activité et un changement d'un ou plusieurs bits de données sur lesquelles nous opérons. Nous avons également analysé des chaînes d'opérations nécessaires pour effectuer des opérations telles que $2P$ ou $P + Q$. Les résultats de l'analyse de chaîne d'operations nécessaires pour effectuer $2P$ pour des versions originales et sécurisées sont présentés dans la figure 6.

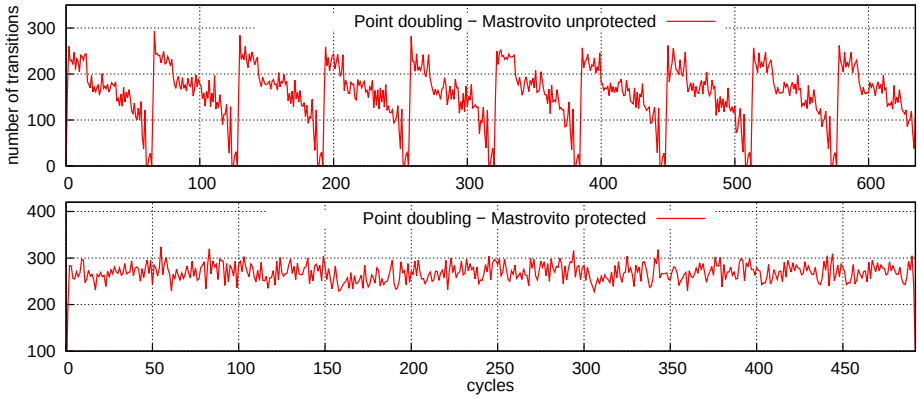

FIGURE 6.: L'activité utile pour l'opération $2P$ réalisée par l'opérateur original ou sécurisée

De plus, pour évaluer des contre-mesures, nous avons utilisé les outils du domaine de traitement du signal (FFT). En utilisant la transformée de Fourier rapide, nous avons transformé des données obtenues dans le domaine temporel vers le domaine fréquenciel. Après nous avons calculé la mesure de planéité spectrale (SFM) [20]. La figure 7 présente les valeurs obtenues. Plus la valeur de SFM s'approche de 1 plus la représentation spectrale de l'activité s'atténue. En observant ces résultats, il est possible de remarquer une amélioration significative de l'uniformité de la distribution d'énergie pour la solution sécurisée basée sur Mastrovito.

Comparaison des courbes d'activité obtenues avec des mesures de courant La dernière étape de la vérification de sécurité était de comparer des courbes d'activité obtenue avec des mesures de courant. Nos différents multiplieurs ont été implémentés dans un circuit Virtex-II sur une carte SASEBO-G (*side-channel attack standard evaluation board*). En utilisant un oscilloscope rapide LeCroy Waverunner 104Xi-a et sonde Tektronix CT1, nous avons mesuré le courant instantané qui est utilisé par le circuit pendant la multiplication. Pour éliminer le bruit autant que possible, nous avons utilisé une alimentation à faible bruit HP E3610A. La figure 8 montre une comparaison entre les mesures obtenues à partir de moniteurs d'activité et les mesures de courant. Les deux premières figures (haut) montrent les résultats pour l'opérateur non sécurisé. Les deux autres figures montrent les résultats pour l'opérateur sécurisé.

L'analyse des résultats permet de conclure que la méthode proposée pour évaluation de l'activité (moniteurs d'activité) reflète bien l'activité "réelle" (courant mesuré dans le circuit). Dans la table 10, nous présentons les résultats d'implementation obtenus pour des opérateurs sécurisés. Pour faciliter la comparaison des opérateurs, nous avons introduit le coefficient α : $\texttt{sécurisé} = \alpha \times \texttt{non-sécurisé}$.

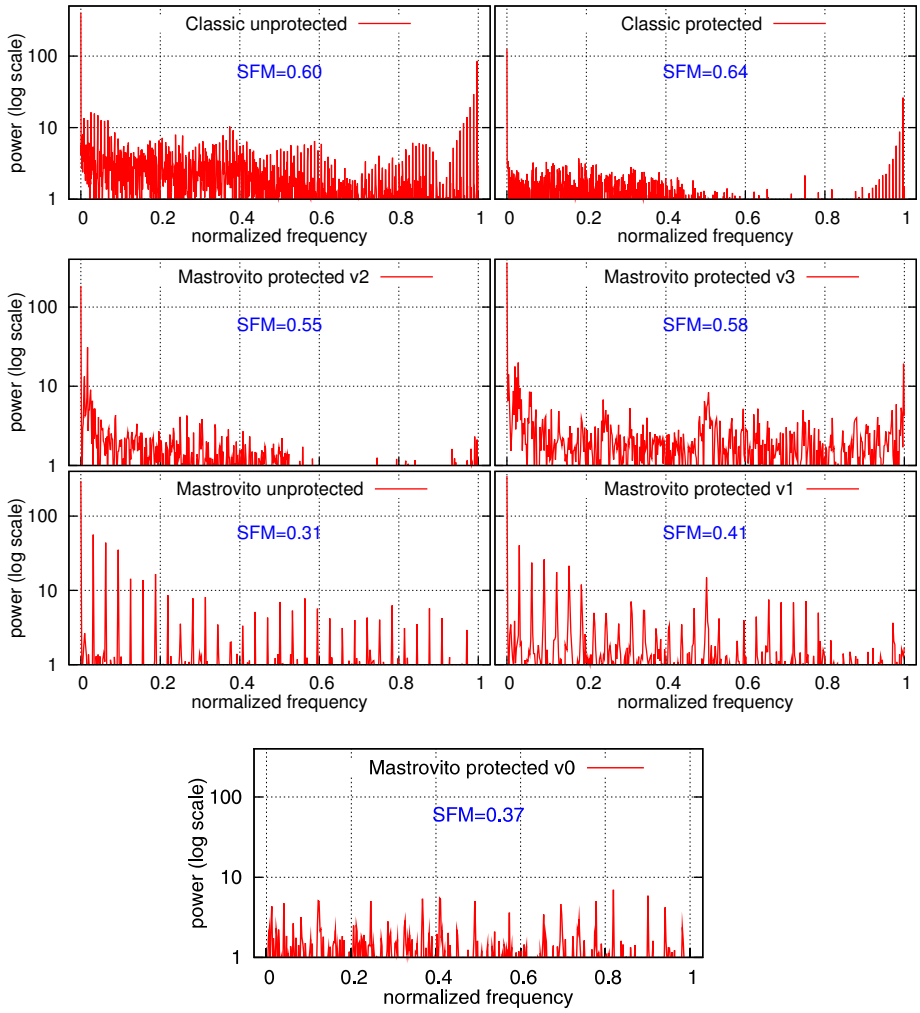

FIGURE 7.: Analyse FFT d'activité utile des opérateurs originaux et sécurisés

4. Résumé et conclusions

L'objectif de cette recherche était de proposer des opérateurs arithmétiques matériels pour le corps fini $GF(2^m)$ efficaces et résistants à certaines

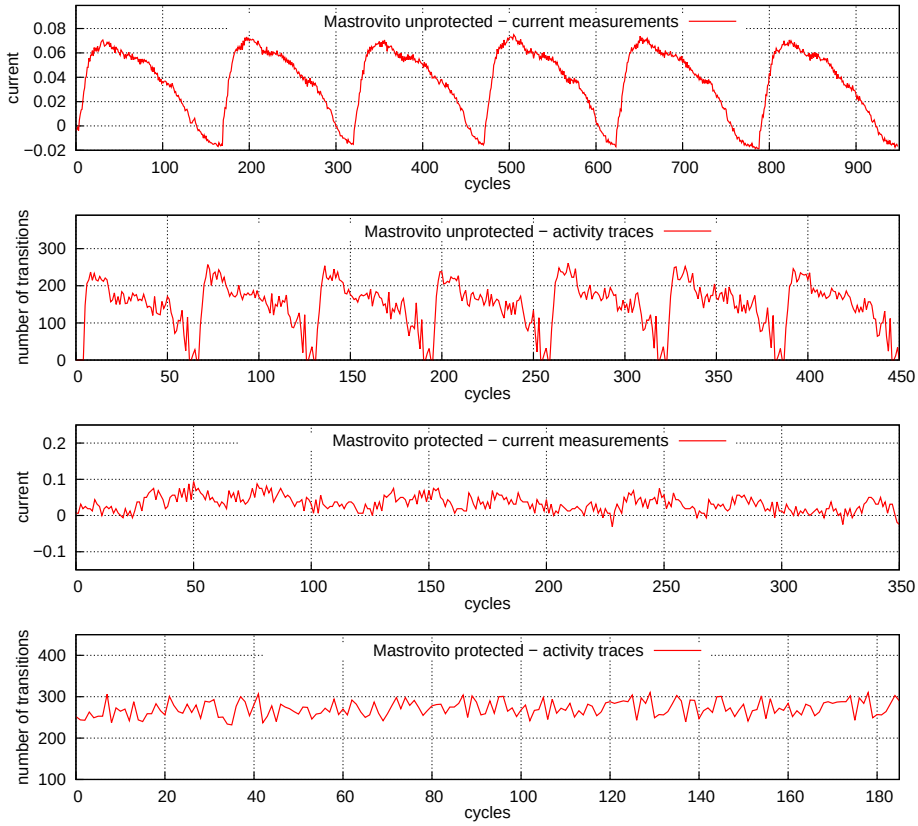

FIGURE 8.: Comparaison d'activite utile et les mesures de courant pour solution basée sur Mastrovito :
pour des versions non sécurisée et sécurisée (uniformisation)

attaques par canaux cachés de type analyse de consommation d'énergie.

Le premier objectif était de développer des opérateurs arithmétiques sur $GF(2^m)$ efficaces, dédiés aux circuits reconfigurables (FPGA) et des applications cryptographiques (systèmes ECC). Afin de réaliser cet objectif, une analyse approfondie des solutions existantes, pendant laquelle nous avons implémenté et vérifié plusieurs propositions d'optimisation, a été effectuée. Il a été supposé que les opérateurs doivent être rapides, de petite taille (occuper une quantité raisonnable de ressources matérielles) et doivent pou-

TABLE 10.: Les résultats d'implémentation des opérateurs sécurisés

Algorithme	ressources LUT ($\times\alpha$)	f MHz ($\times\alpha$)	cycles d'horloge ($\times\alpha$)
Classique	2868 ($\times$0.79)	270 ($\times$0.89)	260 ($\times$0.98)
Montgomery	2099 ($\times$0.96)	323 ($\times$1.00)	264 ($\times$0.98)
Mastrovito v0	3889 ($\times$1.04)	225 ($\times$0.75)	48 ($\times$0.64)
Mastrovito v1	3463 ($\times$1.09)	414 ($\times$1.39)	75 ($\times$1.00)
Mastrovito v2	3700 ($\times$1.02)	306 ($\times$1.03)	avg. 116 ($\times$1.55)
Mastrovito v3	3903 ($\times$1.03)	319 ($\times$1.07)	avg. 80 ($\times$1.07)

voir traiter des très grands nombres (150–600 bits). Nos recherches nous ont permis de trouver, combiner et d'améliorer les caractéristiques des solutions existantes qui permettent de concevoir des opérateurs conformes aux exigences supposées.

Le deuxième objectif était de sécuriser les opérateurs arithmétiques développés contre certaines attaques par canaux cachés de type analyse de consommation d'énergie. Selon les sources connues de la littérature, notre tentative est la première pour sécuriser les opérations au niveau d'opération le plus bas de l'ECC, les opérations sur des éléments des corps finis, sur laquelle les courbes utilisées sont définies. Jusqu'à présent, seulement des protections pour sécuriser les niveaux supérieurs d'opérations des systèmes ECC, telles que les opérations sur les points de courbes elliptiques, ont été proposée dans la litérature.

Les sources potentielles de fuites d'information ont été identifiées de deux manières : par l'utilisation de moniteurs d'activité et d'outil ChipScope et en mesurant le courant instantané consommé par le dispositif à l'aide d'une carte SASEBO-G et d'un équipement spécialisé. À partir de l'analyse des résultats obtenus, il a été conclu que les courbes d'activité de solutions non sécurisés ont des formes très spécifiques. Ces formes spécifiques peuvent permettre d'identifier des opérations sur les points des courbes elliptiques,

comme $2P$ ou $P + Q$. Afin de réduire les fuites d'information, nous avons introduit/proposé des modifications structurelles et des algorithmiques utilisés dans les opérateurs. Des modifications ont été effectuées à très bas niveau pour éviter la dégradation des performances des opérateurs.

Le troisième objectif poursuivi en parallèle avec le deuxième était de trouver un compromis entre la performance et la sécurité des opérateurs. Si des protections avaient eu un impact très négatif sur l'efficacité des opérateurs, il aurait été nécessaire de les modifier ou de les abandonner. La table 10 présente des paramètres des opérateurs protégés.

Pour conclure, à la suite de nos recherches, nous avons atteint les résultats originaux suivants :

- des opérateurs arithmétiques sur $GF(2^m)$ efficaces en matériel dédiés aux systèmes ECC ont été proposés ;
- des moyens pour sécuriser ces opérateurs arithmétiques contre certaines attaques par canaux cachés de type analyse de consommation ont été proposés ;
- un compromis entre l'efficacité des opérateurs et l'efficacité des contremesures a été trouvé ;

Les résultats obtenus ont confirmé la validité de la thèse formulée.

Bibliographie

[1] Ch. W. Chiou, J.-M. Lin, Ch.-Y. Lee, and Ch.-T. Ma. Novel Mastrovito Multiplier over $GF(2^m)$ Using Trinomial. In *Proc. 5th International Conference on Genetic and Evolutionary Computing*, ICGEC '11, pages 237–242, Washington, DC, USA, 2011. IEEE Computer Society.

[2] Xilinx Corporation. Virtex-6 family overview (product specification), 2012.

[3] F. Crowe, A. Daly, and W. Marnane. A scalable dual mode arithmetic unit for public key cryptosystems. In *Information Technology : Coding and Computing (ITCC)*, volume 1, pages 568–573, April 2005.

[4] S. S. Erdem, T. Yanik, and C. K. Koc. Polynomial Basis Multiplication over GF(2^m). *Acta Applicandae Mathematicae*, 93(1-3) :33–55, September 2006.

[5] E. Ferrer, D. Bollman, and O. Moreno. A fast finite field multiplier. In *Proc. 3rd International Conference on Reconfigurable Computing : Architectures, Tools and Applications*, ARC'07, pages 238–246. Springer, 2007.

[6] A. P. Fournaris and O. Koufopavlou. Applying systolic multiplication-inversion architectures based on modified extended Euclidean algorithm for $GF(2^k)$ in elliptic curve cryptography. *Computers & Electrical Engineering, Elsevier*, 33(5-6) :333–348, September 2007.

[7] P. Gallagher. FIPS PUB 186-3 Federal Information Processing Standards Publication Digital Signature Standard (DSS), 2009.

[8] C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, and J. Shokrollahi. FPGA designs of parallel high performance $GF(2^{233})$ multipliers [cryp-

tographic applications]. In *Proc. International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 268–271, May 2003.

[9] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer, 2004.

[10] A. Karatsuba and Y. Ofman. Multiplication of Multi-Digit Numbers on Automata (in Russian). *Doklady Akad. Nauk SSSR*, 145(2) :293–294, 1962. Translation in Soviet Physics-Doklady, 44(7), 1963, p. 595-596.

[11] C. K. Koc and T. Acar. Montgomery Multiplication in GF(2^k). *Designs, Codes and Cryptography*, 14(1) :57–69, April 1998.

[12] H. Li, J. Huang, P. Sweany, and D. Huang. FPGA implementations of elliptic curve cryptography and Tate pairing over a binary field. *Journal of Systems Architecture, Elsevier*, 54(12) :1077–1088, December 2008.

[13] S. Mangard, E. Oswald, and T. Popp. *Power Analysis Attacks : Revealing the Secrets of Smart Cards*. Springer, 2007.

[14] P. L. Montgomery. Modular Multiplication Without Trial Division. *Mathematics of Computation*, 44(170) :519–521, April 1985.

[15] A. H. Namin, W. Huapeng, and M. Ahmadi. Comb Architectures for Finite Field Multiplication in $F(2^m)$. *IEEE Transactions on Computers*, 56(7) :909–916, July 2007.

[16] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Multiplication in $GF(2^m)$: area and time dependency/efficiency/complexity analysis. In *Proceedings of 10th International IFAC Workshop on Programmable Devices and Embedded Systems (PDES)* , pages 43–48, 2010.

[17] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Analiza algorytmów mnozenia w ciele $GF(2^m)$. *Pomiary Automatyka Kontrola (PAK) (Measurement, Automation and Monitoring)* , 57(01/2011) :58–60, 2011.

[18] D. Pamula, E. Hrynkiewicz, and A. Tisserand. Analysis of $GF(2^m)$ multipliers regarding Elliptic Curve Cryptosystem applications. In

Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems (PDES) , pages 252–257, 2012.

[19] D. Pamula and A. Tisserand. $GF(2^m)$ Finite-Field Multipliers with Reduced Activity Variations. *WAIFI 2012, LNCS 7369. Springer*, pages 152–167.

[20] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing.* Prentice Hall, 1996.

[21] S. Radack. Guide to protecting personally identifiable information. ITL Bulletin for April 2010.

[22] Research Center for Information Security National Institute of Advanced Industrial Science and Technology. Side-channel Attack Standard Evaluation Board. SASEBO-G. Specification. Version 1.0 , 2008.

[23] F. Rodríguez-Henríquez, N. A. Saqib, and A. Díaz-Pérez. A fast parallel implementation of elliptic curve point multiplication over GF(2^{m}). *Microprocessors and Microsystems*, 28(5-6) :329–339, 2004.

[24] A. Tisserand. Fast and Accurate Activity Evaluation in Multipliers. In *Proc. 42nd Asilomar Conference on Signals, Systems and Computers*, pages 757–761, Pacific Grove, California, U.S.A., October 2008. IEEE.

[25] S. Vanstone. ECC holds key to next generation cryptography. *[Online].Available : http ://www.design-reuse.com/articles/7409/ecchold-key-to-next-gen-cryptography.html*, March 2006.

[26] J. Wang and A. Jiang. A high-speed dual field arithmetic unit and hardware implementation. In *ASIC, 2007. ASICON '07. 7th International Conference on*, pages 213–216, October 2007.

[27] H. Wu, M. A. Hasan, I. F. Blake, and S. Gao. Finite field multiplier using redundant representation. *IEEE Transactions on Computers*, 51(11) :1306–1316, November 2002.

[28] Xilinx Corporation. Virtex-II Pro and Virtex-II Pro X Platform FPGAs : Complete Data Sheet (Product Specification), 2007.

[29] Xilinx Corporation. Spartan-3E FPGA Family : Data Sheet (Product Specification), 2009.