SympA'6
Besançon, 19 - 22 juin 2000
6ème Symposium sur les Architectures Nouvelles de Machines

Une comparaison entre quelques implantations logicielles et matérielles de l'algorithme de chiffrement IDEA

Jean-Luc Beuchat[1], Jacques-Olivier Haenni[1], Christof Teuscher[1], Francisco J. Gómez[2], Hector Fabio Restrepo[1] et Eduardo Sanchez[1]

[1]Laboratoire de Systèmes Logiques, Ecole Polytechnique Fédérale de Lausanne
CH – 1015 Lausanne, Suisse
E-mail : {name.surname}@di.epfl.ch
Web : http://lslwww.epfl.ch

[2]Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid
E – 20849 Madrid, Spain
E-mail : francisco.gomez@ii.uam.es

Résumé

Dans cet article, nous décrivons et comparons quelques implantations récentes, tant logicielles que matérielles, de l'algorithme de cryptage IDEA. Nous nous attardons plus particulièrement sur deux prototypes réalisés dans notre laboratoire. Le premier exploite le jeu d'instructions multimédia d'Itanium, le nouveau processeur d'Intel et Hewlett Packard. Le second, appelé *CryptoBooster*, est un coprocesseur cryptographique implanté sur un circuit FPGA. Parmi les solutions étudiées, celle offrant le meilleur débit est matérielle. Nous avons toutefois obtenu de très bons débits (de l'ordre de 1.5 Gbits/s) en simulant du code optimisé pour Itanium. La bande passante ne constitue pas l'unique critère de comparaison de dispositifs de chiffrement. La sécurité garantie par des systèmes matériels ou logiciels n'est par exemple pas équivalente.

1 Introduction

IDEA, un algorithme de chiffrement de données par blocs, offre d'excellentes garanties de sécurité. A ce jour, personne n'a encore publié de résultats démontrant des faiblesses de l'algorithme. IDEA constitue ainsi un choix judicieux pour le cryptage de transmissions de données. Les protocoles de communication offrant des débits de plus en plus élevés, un simple programme C ne peut pas satisfaire les critères temps réel requis et plusieurs groupes de recherche ont proposé des solutions logicielles ou matérielles plus subtiles. Cet article propose une étude de quelques systèmes décrits dans des publications récentes.

Après une brève description de l'algorithme IDEA (section 2), nous présentons quelques implantations intéressantes en nous attardant plus particulièrement sur deux systèmes développés dans notre laboratoire (section 3). Nous effectuons ensuite une comparaison des solutions étudiées (section 4).

2 L'algorithme IDEA

IDEA [1] (*International Data Encryption Algorithm*) est un des algorithmes de chiffrement de données par blocs proposés ces dernières années afin de remplacer DES. Xuejia Lai et James Massey [7], deux chercheurs de l'Ecole Polytechnique Fédérale de Zurich, ont conçu une première version de cet algorithme, appelée PES (*Proposed Encryption Standard*), en 1990. Suite aux travaux de Biham et Shamir concernant la cryptanalyse différentielle, ils ont renforcé leur système contre les attaques et l'ont baptisé IPES (*Improved Proposed Encryption Standard*), puis IDEA en 1992. Huit rondes de calcul et une ronde finale

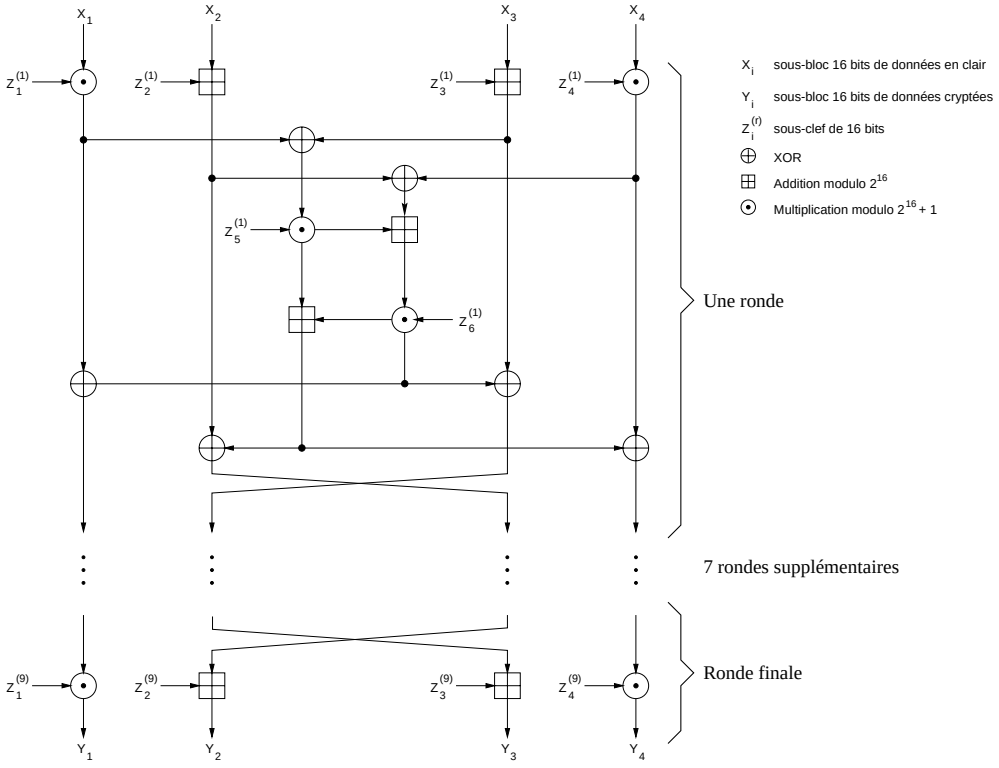

FIG. 1 - *Schéma du processus de cryptage.*

constituent le système de chiffrement (figure 1). Trois types d'opérations interviennent dans le processus de cryptage : le OU exclusif bit à bit de deux mots de 16 bits, l'addition entière 16 bits modulo 2^{16} et la multiplication entière 16 bits modulo $2^{16}+1$. Un mot X de 64 bits est partitionné en quatre sous-blocs X_1, X_2, X_3 et X_4 de 16 bits chacun. Ces derniers sont ensuite transformés en quatre sous-blocs cryptés Y_1, Y_2, Y_3 et Y_4 en fonction de 52 sous-clefs de 16 bits déterminées à partir d'une clef de 128 bits. Le décryptage se déroule de manière analogue. L'unique différence réside dans le calcul des sous-clefs.

Afin de sécuriser une transmission de données, la première solution venant à l'esprit consiste à crypter (par exemple avec IDEA) et envoyer chacun des blocs d'un message.

[1] IDEA est protégé par un brevet appartenant à Ascom en Europe, aux Etats-Unis et au Japon. Toute application commerciale nécessite par conséquent l'achat d'une licence. Son utilisation est néanmoins gratuite dans le cadre de projets académiques ou à but non lucratif.

Cette approche, appelée *electronic codebook* (*ECB*), permet de traiter indépendamment les différentes parties d'un texte. En effet, pour une clef spécifiée, à chaque bloc de données en clair correspond un unique bloc de données cryptées. Supposons maintenant qu'un espion dispose de quelques messages en clair et des messages cryptés associés. Il peut ainsi construire son propre *codebook*, puis décrypter partiellement de nouveaux messages. Il lui est également possible de modifier des informations cryptées sans connaître la clef [11].

Les modes de chaînage apportent une solution simple aux problèmes de sécurité du mode *ECB*. Grâce à un mécanisme de rétroaction, un bloc crypté est utilisé pour modifier le chiffrement du prochain bloc. *Cipher block chaining* (*CBC*), l'un des modes de chaînage les plus simples, effectue un OU exclusif entre un bloc de données en clair et un bloc crypté. Remarquons toutefois que la même information résulte du chiffrement de deux messages identiques. L'utilisation d'un vecteur initial, généré aléatoirement et combiné avec le premier bloc, pallie cet inconvénient.

L'écriture d'un programme dans un langage de haut niveau est certainement la méthode la plus simple pour implanter IDEA. Ascom distribue gratuitement le code C de l'algorithme[2] et propose une comparaison des performances obtenues avec divers processeurs commerciaux[3] (tableau 1). Aucun de ces systèmes ne permet d'atteindre le débit requis par un protocole tel *Gigabit Ethernet*. D'autres approches sont par conséquent indispensables afin de crypter des données en temps réel à haut débit.

Processeur	**Débit [kbits/s]**
VAX 8650	430
486DX2-66	1'700
Pentium, 90 MHz	4'600
PentiumPro, 180 MHz	16'000

TAB. 1 - *Performances du code C d'IDEA distribué par Ascom.*

La réalisation d'un coprocesseur spécifique à l'algorithme IDEA est une alternative aux solutions logicielles. Nous distinguons essentiellement deux technologies permettant la réalisation de tels circuits : les ASIC et les FPGA. Nous invitons le lecteur ne connaissant pas ces types de circuit à consulter [12].

La multiplication modulo $2^{16}+1$ constitue peut-être la principale difficulté de l'implantation d'IDEA. Une méthode utilisée aussi bien en logiciel qu'en matériel pour effectuer ce calcul est décrite par l'équation ci-dessous [5] :

$$xy \bmod (2^n+1) = (xy \bmod 2^n - xy \operatorname{div} 2^n) \bmod (2^n+1). \tag{1}$$

Si x et y sont deux nombres strictement positifs de n bits, cette équation peut également s'écrire de la manière suivante :

$$xy \bmod (2^n+1) = \begin{cases} xy \bmod 2^n - xy \operatorname{div} 2^n + 2^n + 1 & \text{si } xy \operatorname{div} 2^n > xy \bmod 2^n, \\ xy \bmod 2^n - xy \operatorname{div} 2^n & \text{sinon.} \end{cases} \tag{2}$$

[2] http://www.ascom.com/infosec/downloads.html
[3] http://www.ascom.com/infosec/idea/benchmark.html

3 Quelques implantations de l'algorithme IDEA

3.1 Solutions logicielles

Un des moyens mis en œuvre pour augmenter les performances des processeurs à usage général est l'exploitation du parallélisme intrinsèque des programmes. Ces processeurs en tirent parti à deux niveaux :

- **Parallélisme au niveau des instructions** (ILP, *Instruction-Level Parallelism*). Les processeurs superscalaires, VLIW (*Very Large Instruction Word*) et EPIC (*Explicitly Parallel Instruction Computing*) possèdent plusieurs unités de traitement leur permettant d'exécuter plusieurs instructions différentes en même temps.

- **Parallélisme de type SIMD** (*Single-Instruction, Multiple-Data*). La plupart des processeurs actuels disposent d'un jeu d'instructions multimédia (ou SIMD), permettant de considérer un registre comme un ensemble de plusieurs valeurs 8 ou 16 bits. Ces instructions appliquent la même opération à toutes les valeurs d'un registre (traitement vectoriel). A ce jour, aucun compilateur n'est capable d'exploiter ce type de parallélisme.

Parmi les nombreuses implantations logicielles d'algorithmes de cryptage, notons les travaux de Biham [1], qui a implanté l'algorithme DES sur un processeur Alpha en considérant ce dernier comme une machine SIMD 64x1 bit, de Lipmaa [8], qui a codé IDEA sur les processeurs Pentium et Pentium II en exploitant leurs instructions MMX, et de Clapp [3], qui a étudié l'implantation de routines cryptographiques sur des processeurs VLIW, SIMD et superscalaires.

La suite de ce chapitre présente une implantation d'IDEA sur Itanium. Cette implantation a été réalisée dans notre laboratoire.

3.1.1 Implantation sur Itanium

IA-64 (Intel Architecture 64) [6] est une nouvelle famille de processeurs développée par Intel et HP. Il s'agit d'une architecture 64 bits de type EPIC (Explicitly Parallel Instruction Computing). Le premier représentant de cette famille est Itanium (nom de code Merced) qui est capable d'exécuter jusqu'à six instructions en parallèle. Il est muni d'un jeu d'instructions multimédia qui permet de considérer un registre 64 bits comme contenant deux valeurs 32 bits, quatre valeurs 16 bits ou huit valeurs 8 bits.

Les performances de ce processeur seront donc maximales dans le cas où toutes les unités de traitement sont utilisées (i.e. parallélisme au niveau des instructions) et que les unités de calcul sont pleinement utilisées (i.e. traitement de valeurs 64 bits).

Dans le cas de l'algorithme de cryptage IDEA, le parallélisme intrinsèque du programme est relativement faible et les valeurs manipulées tiennent sur 16 bits. La suite de cette section montre comment ces deux problèmes sont abordés.

Parallélisme de type SIMD

Pour exploiter ce type de parallélisme, il faut identifier des instructions identiques pouvant s'exécuter en parallèle. De telles instructions se trouvent, par exemple, à la fin de chaque ronde (quatre OU exclusif indépendants) ou au début d'une ronde. Ces instructions se prêtent mal à une implantation à l'aide d'instructions multimédia car cela impli-

querait un grand travail de réorganisation des données entre chaque instruction. De plus, ce parallélisme est très limité.

Afin d'augmenter le parallélisme disponible, nous allons considérer le cryptage en parallèle de quatre mots de 64 bits successifs X^1, X^2, X^3 et X^4. Ces calculs sont indépendants si nous utilisons les modes de chaînage *ECB* ou *CBC* avec 8 vecteurs initiaux.

Ainsi, chaque opération de l'algorithme IDEA se retrouve lors du cryptage de chacun de ces quatre mots, ce qui nous permet d'utiliser des instructions multimédia manipulant des vecteurs de quatre valeurs de 16 bits. Cette partie du code bénéficiera donc d'un *speed-up* d'environ quatre.

Le seul *overhead* induit par cette implantation consiste en deux transpositions de matrices 4x4 (X_i^j et Y_i^j, $i, j = 1, .., 4$) avant et après le cryptage afin de regrouper les mots de 16 bits qui sont traités en parallèle.

Parallélisme de type ILP

Le parallélisme intrinsèque de IDEA est assez restreint, et il nous a permis d'obtenir, en moyenne, une exécution de 3.5 à 3.9 instructions par coup d'horloge dans la boucle de cryptage.

Afin d'augmenter ce parallélisme, nous proposons de ne pas crypter quatre mots en parallèle, mais huit. Cela double le niveau de parallélisme présent dans notre code et nous a permet l'augmentation du nombre moyen d'instructions exécutées par coup d'horloge à 5.4.

Implantation

Les trois opérations de base. L'addition modulo 2^{16} et le OU exclusif sont directement implantés à l'aide des instructions correspondantes d'Itanium. La multiplication modulo $2^{16} + 1$ est réalisée à l'aide d'une variante de l'algorithme présenté à la section 2. Nous sommes parvenus à coder cette opération en 14 instructions nécessitant 7 coups d'horloge.

Gestion des clefs. Afin de simplifier la routine de cryptage, les 52 sous-clefs sont calculées au préalable et sont stockées dans des registres du processeur.

Accès à la mémoire. Les accès à la mémoire sont les opérations les plus pénalisantes de notre programme de par leur latence pouvant s'avérer très élevée. Les seuls accès qui ont lieu, une fois le programme initialisé, sont les accès de lecture des données à crypter et d'écriture des données cryptées. Afin de cacher au mieux les latences des lectures, nous avons décomposé la boucle qui parcourt les données à traiter en deux étapes (chargement et traitement des données) et nous l'avons implantée à l'aide de la méthode appelée "*software pipelining*" en chargeant des données qui ne seront utilisées qu'à l'itération suivante. Ainsi, le chargement des données et leur utilisation sont séparés d'environ 260 coups d'horloge.

Performances

Ce logiciel n'a pas été testé sur un circuit physique. Comme les latences des instructions d'Itanium ne sont pas encore publiques, nous avons utilisé celles du Pentium III fournissant une bonne approximation. Nous avons ainsi compté un cycle pour toutes les instructions arithmétiques entières, à l'exception de la multiplication entière (trois cycles).

La routine de cryptage est écrite en assembleur; il est donc facile d'estimer les performances du système. De plus, la fonctionnalité du code a été vérifiée par simulation.

Ainsi, nous estimons actuellement que ce programme sera capable de crypter ou décryp-

ter des données stockées en mémoire à un débit de 1.5 Gbit/s sur un processeur Itanium à 800 MHz. Ce chiffre devra toutefois être vérifié sur une machine physique, et nous craignons que les résultats ne soient inférieurs, car (1) il est très difficile d'estimer l'influence des accès mémoire sur les performances, et (2) nous supposons que notre code est le seul programme exécuté par le processeur, ce qui n'est pas le cas dans la réalité.

3.2 Solutions matérielles

Nous présentons brièvement dans ce paragraphe quelques implantations matérielles d'IDEA réalisées par différents groupes de chercheurs. Nous nous attardons plus particulièrement sur un prototype réalisé dans notre laboratoire ainsi que sur sa nouvelle version actuellement en cours de développement.

Le circuit intégré VINCI

En 1993, une équipe de l'Ecole Polytechnique Fédérale de Zurich a développé un circuit intégré implantant IDEA [4, 15]. Le *chip*, réalisé dans une technologie 1.2 μm, contient 251'000 transistors pour une superficie totale de 107.8 mm^2. Offrant un débit de 177.8 Mbits/s (@ 25 MHz), VINCI fut le premier système capable de crypter des données en temps réel pour des protocoles comme ATM (*Asynchronous Transfer Mode*) ou FDDI (*Fiber Distributed Data Interface*).

Le coprocesseur IDEACrypt d'Ascom

Ascom, propriétaire des brevets IDEA, propose une implantation de l'algorithme sous la forme d'un noyau ASIC (technologie 0.25μm) embarqué appelé IDEACrypt. Destiné aux applications commerciales, IDEACrypt offre des caractéristiques intéressantes du point de vue de la sécurité. Le circuit dispose d'une mémoire non volatile de 32 clefs dont la lecture est impossible de l'extérieur. Il est possible de crypter les clefs des sessions à l'aide d'une clef maîtresse avant de les envoyer à IDEACrypt. Le circuit dispose de cette clef maîtresse et peut ainsi décrypter les clefs de sessions, puis calculer les sous-clefs associées.

IDEACrypt propose les modes de chaînage les plus courants (ECB, CBC, ...). A une fréquence d'horloge de 40 MHz, ce dispositif crypte 300 Mbits/s en mode ECB et 100 Mbits/s dans les autres modes.

IDEA sur un FPGA XC4005 de Xilinx

Les *benchmarks* destinés à l'évaluation des performances d'un FPGA sont constitués de quelques circuits très simples comme des compteurs 16 bits ou de petites machines d'états. Suite à cette constatation, Caspi et Weaver ont suggéré l'utilisation d'IDEA [2]. Cet algorithme est en effet relativement simple à implanter et offre une métrique (débit en Mbits/s) permettant une comparaison aisée de différents circuits. Le système décrit dans [2] fonctionne sur un FPGA XC4005 de Xilinx. La faible taille de ce circuit interdit l'implantation d'une ronde complète et explique la faible bande passante de 0.447 Mbits/s (@ 7.3 MHz).

Une comparaison entre des solutions DSP et FPGA

En 1998, Mencer, Morf et Flynn ont réalisé deux implantations d'IDEA, l'une sur un DSP de Texas Instruments (TI TMX320C6x), l'autre sur des FPGA de la famille XC4000

de Xilinx [9]. Le DSP comporte deux multiplicateurs, quatre ALU et offre une architecture VLIW d'une largeur de quatre opérations. Le système obtenu crypte les données à 53.1 Mbits/s (@ 200 MHz). Mencer *et al.* proposent également une implantation distribuée sur quatre FPGA XC4000 XL de Xilinx des huit rondes et de l'étage final d'IDEA. Cette solution offre une bande passante de 528 Mbits/s (@ 33 MHz).

PipeRench

PipeRench [13] est une structure reconfigurable sous forme de *pipeline* supportant la virtualisation matérielle. Ceci permet d'implanter des *designs* dépassant les ressources physiques du circuit. Chaque étage du *pipeline* est composé d'éléments de calcul simples, similaires aux cellules d'un FPGA. Un compilateur très rapide génère les configurations pour le circuit. La simulation d'un circuit en technologie 0.25 μm offre une bande passante de 126.6 Mbits/s (@ 100 MHz), c'est-à-dire 40% de plus que IDEACrypt. Une première réalisation en technologie 0.35 μm du *chip* était prévue pour fin 1999.

CryptoBooster

Développé dans notre laboratoire dans le cadre d'un projet industriel, *CryptoBooster* est un coprocesseur cryptographique destiné aux implantations sur circuits FPGA [10]. Il est connecté à un système hôte lui envoyant les paquets à crypter ainsi qu'à une mémoire organisée en sessions contenant chacune un ensemble de paramètres tels la clef de cryptage ou les vecteurs initiaux. Lorsqu'un paquet est transmis à *CryptoBooster*, l'hôte précise quelle session utiliser. Notre coprocesseur offre évidemment un ensemble de commandes permettant la création, la modification ou la lecture des sessions. *CryptoBooster* possède une architecture modulaire (figure 2). Cette approche permet de ne modifier qu'une partie du système afin d'implanter un nouvel algorithme de cryptage ou un nouveau mode de chaînage. Les modules ***HostInterface*** et ***MemAdapter*** gèrent l'échange de données entre le coprocesseur et son environnement. Ils sont respectivement spécifiques à un protocole de communication et à un type de mémoire. Décomposé en plusieurs blocs, le module ***CryptoCore*** constitue l'unité de calcul du système. ***SessionAdapter*** s'occupe de la gestion des paramètres des sessions et du calcul des sous-clefs. ***CypherCore*** contient l'algorithme de cryptage et le mode de chaînage. Finalement, le module ***SessionControl*** orchestre les transferts de données entre le système hôte, la mémoire et le *CryptoCore*.

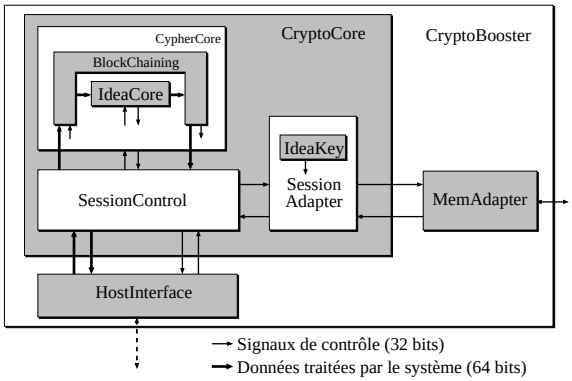

FIG. 2 - *Architecture générale du coprocesseur cryptographique.*

Le premier *CypherCore* réalisé propose l'algorithme IDEA décrit brièvement au paragraphe 2. L'implantation des huit rondes de calcul et de l'étage final nécessite cependant beaucoup de ressources matérielles et n'est envisageable que sur les plus gros circuits FPGA actuellement disponibles. Nous avons par conséquent adopté une solution très flexible permettant le choix du nombre de rondes m à la compilation. Comme les huit rondes sont

identiques, il est en effet possible d'utiliser itérativement $n = \frac{8}{m}$ fois un bloc comportant m rondes, $m \in \{1, 2, 4, 8\}$.

Afin de tester le système, nous avons utilisé une carte *RC1000–PP*, commercialisée par Embedded Solutions Limited [4], proposant un FPGA Virtex XCV1000 de Xilinx. A une fréquence de 20 MHz, nous obtenons un débit de $160 \cdot m$ Mbits/s, où m dénote le nombre de rondes apparaissant sur le *chip*. La taille du FPGA nous permet d'implanter une seule ronde.

Vers une nouvelle version de *CryptoBooster*

Les performances de *CryptoBooster* ne permettent pas le traitement de données en temps réel pour des communications à très haut débit. *CryptoBooster* effectue la multiplication modulo $2^{16} + 1$ selon l'équation 2. Plusieurs chercheurs ont proposé des alternatives à cet algorithme. En nous inspirant d'une méthode proposée par Zimmermann [14], nous avons développé un multiplicateur modulo $2^{16} + 1$, de latence égale à deux, optimisé pour la famille Virtex et fonctionnant à une fréquence de 50 MHz. Nous espérons implanter prochainement les huit rondes et l'étage final d'IDEA sur un FPGA tournant à 50 MHz. Nous obtiendrons ainsi un système, appelé provisoirement *CryptoBooster* II, offrant une bande passante de 3.2 Gbits/s.

4 Comparaison des différentes implantations d'IDEA

Ce paragraphe propose une brève comparaison des diverses implantations d'IDEA présentées dans cet article. Les tableaux 2 et 3 résument leurs principales caractéristiques. Rappelons toutefois que les débits mesurés pour Itanium et *CryptoBooster* II résultent uniquement de simulations. Nous ne donnons donc ces chiffres qu'à titre indicatif.

Processeur	**Remarque**	**Débit [Mbits/s]**	**Fréquence [MHz]**
Pentium	Sans utilisation des instructions MMX	4.6 (Ascom)	90
PentiumPro	Sans utilisation des instructions MMX	16 (Ascom)	180
Pentium MMX		28.2 - 28.5 [8]	166
Pentium MMX	Exécution de 4 cryptages en parallèle	71.8 - 72.4 [8]	166
Pentium II		32.2 - 32.9 [8]	233
Pentium II	Exécution de 4 cryptages en parallèle	105.1 - 107.2 [8]	233
Itanium	Exécution de 4 cryptages en parallèle	*870*	800
Itanium	Exécution de 8 cryptages en parallèle	*1500*	800

TAB. 2 - *Performances des implantations logicielles. Les débits en italique résultent de simulations.*

Le système exploitant quatre XC4000 XL, développé par Mencer *et al.* [9], offre la bande passante la plus élevée. Il semblerait toutefois qu'une nouvelle version de *PipeRench* atteigne un débit supérieur à 1 Gbits/s. A notre connaissance, ce résultat, annoncé informellement lors de la conférence *CHES'99*, n'a pas encore été publié. Actuellement, les

[4] http://www.embedded-solutions.ltd.uk/

Groupe	Circuit	Type	Débit [Mbits/s]	Fréquence [MHz]
Zimmermann *et al.* [4]	VINCI	CMOS 1.2 μm	177.8	25
Ascom	IDEACrypt	CMOS 0.25 μm	100 à 300	40
Caspi et Weaver [2]	XC4005	FPGA	0.477	7.3
Mencer *et al.* [9]	TI TMX320C6x	DSP	53.1	200
Mencer *et al.* [9]	**XC4000 XL**	**FPGA**	**528**	**33**
Taylor *et al.* [13]	*PipeRench*		*126.6*	100
Mosanya *et al.* [10]	Virtex 1000	FPGA	*160*	20
CryptoBooster II	Virtex 1000	FPGA	*3200*	50

TAB. 3 - *Performances des implantations matérielles. Les débits en italique résultent de simulations.*

approches matérielles constituent la meilleure alternative du point de vue de la bande passante. Cette tendance pourrait évidemment changer si les résultats des simulations Itanium se confirment en pratique et si *CryptoBooster* II n'atteint pas les performances escomptées.

Le débit ne constitue cependant pas le seul critère de comparaison. Les aspects de sécurité s'avèrent également cruciaux pour des produits commerciaux. La confidentialité des clefs constitue ainsi un problème majeur. Les approches matérielles offrent une meilleure protection que les solutions logicielles. Rappelons simplement que la lecture des clefs stockées dans le circuit IDEACrypt d'Ascom est impossible de l'extérieur. Nous invitons le lecteur souhaitant plus de détails concernant ce sujet à consulter [11].

5 Conclusion

Les implantations matérielles d'IDEA se révèlent actuellement plus rapides et offrent une meilleure sécurité que les approches logicielles. Toutefois, nous avons montré qu'il est possible d'obtenir de très bonnes performances en logiciel, à condition d'écrire le code directement en assembleur. Les compilateurs ne sont actuellement pas assez performants pour mettre en évidence suffisamment de parallélisme, et, par conséquent, exploiter toutes les ressources des processeurs modernes. De plus, le coût de développement et de mise en service d'un système matériel est plus élevé que pour un système logiciel.

Les systèmes de cryptage logiciels et matériels ont des domaines d'applications très différents allant du cryptage de fichiers sur un ordinateur de bureau au chiffrement de toutes les communications entre deux sites.

Enfin, des systèmes mixtes matériel-logiciel (par exemple, cryptage d'un mot en matériel et implantation du mode de chaînage en logiciel) peuvent également constituer des solutions intéressantes et entrent dans le domaine du *co-design*.

Remerciements

Nous tenons à remercier Hewlett Packard et Lightning [5] qui ont partiellement financé ce travail.

[5] http://www.lightning.ch

Bibliographie

1. E. Biham. A Fast New DES Implementation in Software. In E. Biham, editor, *Fast Software Encryption*, number 1267 in Lecture Notes in Computer Science, pages 260–271. Springer, 1997.
2. E. Caspi and N. Weaver. IDEA as a benchmark for reconfigurable computing. Technical report, BRASS Research Group, University of Berkeley, December 1996. Report available form http://www.cs.berkeley.edu/projects/brass/projects.html.
3. S. K. Clapp. Optimizing a Fast Stream Cipher for VLIW, SIMD, and Superscalar Processors. In E. Biham, editor, *Fast Software Encryption*, number 1267 in Lecture Notes in Computer Science, pages 273–287. Springer, 1997.
4. A. Curiger, H. Bonnenberg, R. Zimmermann, N. Felber, H. Kaeslin, and W. Fichtner. VINCI: VLSI implementation of the new block cipher IDEA. In *Proceedings of the IEEE CICC'93*, pages 15.5.1–15.5.4, San Diego, CA, May 1993.
5. Andreas V. Curiger, Heinz Bonnenberg, and Hubert Kaeslin. Regular VLSI Architectures for Multiplication Modulo $(2^n + 1)$. *IEEE Journal of Solid–State Circuits*, 26(7):990–994, July 1991.
6. Intel. *Intel IA-64 Architecture Software Developer's Manual*, January 2000.
7. X. Lai. *On the Design and Security of Block Ciphers*. Number 1 in ETH Series in Information Processing. Hartung-Gorre Verlag Konstanz, 1992.
8. H. Lipmaa. IDEA: A Cipher for Multimedia Architectures? In S. Tavares and H. Meijer, editors, *Selected areas in cryptography*, number 1556 in Lecture Notes in Computer Science, pages 248–263. Springer, 1999.
9. O. Mencer, M. Morf, and M. J. Flynn. Hardware software tri-design of encryption for mobile communication units. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, Seattle, Washington, USA, May 1998.
10. E. Mosanya, C. Teuscher, H. F. Restrepo, P. Galley, and E. Sanchez. CryptoBooster: A Reconfigurable and Modular Cryprographic Coprocessor. In C. K. Koc and C. Paar, editors, *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, CHES'99*, number 1717 in Lecture Notes in Computer Science, pages 246–256. Springer, August 1999.
11. B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, 2nd edition, 1996.
12. Michael John Sebastian Smith. *Application–Specific Integrated Circuits*. Addison–Wesley, 1997.
13. R. Reed Taylor and Seth Copen Goldstein. A High–Performance Flexible Architecture for Cryptography. In C. K. Koc and C. Paar, editors, *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems, CHES'99*, number 1717 in Lecture Notes in Computer Science, pages 231–245. Springer, August 1999.
14. R. Zimmermann. Efficient VLSI implementation of modulo $(2^n + 1)$ addition and multiplication. In *Proceedings IEEE Symposium on Computer Arithmetic*, Adelaide, Australia, April 1999.
15. R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner. A 177 Mbit/s VLSI implementation of the international data encryption algorithm. *IEEE Journal of Solid-State Circuits*, 29(3):303–307, March 1994.