Smart Card Crypto-Coprocessors for Public-Key Cryptography

[Published in J.-J. Quisquater and B. Schneier, Eds., *Smart Card Research and Applications*, vol. 1820 of *Lecture Notes in Computer Science*, pp. 386–394, Springer-Verlag, 2000.]

Helena Handschuh[1,2] and Pascal Paillier[1,2]

[1] ENST, Computer Science Department
46 rue Barrault, 75634 Paris Cedex, France
[2] Gemplus, Cryptography Department
34 rue Guynemer, 92447 Issy-les-Moulineaux, France
{helena.handschuh, pascal.paillier}@gemplus.com

Abstract. This paper intends to provide information about up-to-date performances of smart-card arithmetic coprocessors regarding major public-key cryptosystems and analyze the main tendences of this developing high-tech industry and related markets. We also comment hardware limitations of current technologies and provide a technique for extending them by virtually doubling their capacities.

1 Introduction

In recent years, public key cryptography [4] has gained increasing attention from both companies and end users who wish to use this technology to securize a wide variety of applications. One consequence of this trend has been the growing importance of public-key smart cards to store the user's private keys and provide a secure computing environment for private key operations. As such, smart cards can be seen as secure tokens.

Many chip manufacturers are therefore proposing ever better and faster implementations of public key algorithms using dedicated crypto-coprocessors on their chips. Two years ago the main focus of attention was on established public-key techniques such as RSA and Discrete Logarithm based signature schemes [14]. Main industrial efforts concentrated on the performance of these algorithms with key sizes that were lying around the 512-bit range. A survey of the state of the art at that time was published in [11]. This paper is intended to give an overview of recent evolutions in the crypto-coprocessor market and the latest achievements in terms of speed and available modulus size ranges. We also include consideration of implementations of other public-key techniques such as elliptic curve signature schemes [9].

2 The Public-Key Battlefield

The most widely used smart card chips with crypto-coprocessors are listed here. The usual manufacturers such as SGS-Thomson, Siemens, Philips and Motorola (now part of Atmel) still lead the market, but new manufacturers have more actively joined the public-key battlefield since 1996, such as NEC, Hitachi and Toshiba. Table 2 and 4 given in appendix list the standard field size of such chips in terms of on-board memory sizes (RAM, ROM, EEPROM), operating voltage and frequency, abnormal behaviour sensors (high or low voltage (HV/LV/V), frequency (HF/LF/F) or temperature (HT/LT/T)), maximum key size supported for RSA or DSA public moduli, and elliptic curve characteristic as appropriate.

It can be observed that new trends include increasing RAM and EEPROM sizes for user customized applications, faster internal clocks, improved security features such as new sensors or address scrambling and security matrices, and growing public modulus sizes. In other words, cryptographic chips are becoming bigger, more versatile, faster and increasingly secure.

The new range of key sizes for RSA and DSA is now generally up to 1024 bits, and some chips even handle 2048-bit computations. But as the need for ever bigger moduli becomes less acute, one future direction for improvement is clearly in a battle for increased performance. Public key algorithms are currently used only for authentication or access control as they are much too slow for stream encryption.

Every architecture presents its own optimizations for computing modular multiplications and exponentiations which are the basic operations in most public key cryptosystems. See [11] for some examples. But basically most chips end up achieving comparable speed. Tables 3 and 5 list the computing times on different platforms regarding various public key algorithms, with the most widely implemented system being RSA. However, DSA [7] and its elliptic curve version ECDSA, as well as other schemes such as ESIGN or zero-knowledge identification techniques [6] appear in some applications.

With regard to RSA signature generation, only 1024-bit implementations run with perfomance times of less than a second for now, but we suspect it will not be too long before 2048-bit RSA becomes practical as well. As a matter of fact, in one of Gemplus' custom implementations, 2048-bit signatures can already be computed in a reasonable time (around 1.5 seconds) using the Chinese Remainder Theorem and a signature verification using a small exponent (typically $e = \mathrm{F}_4 = 2^{16} + 1$) can be computed in about 270 ms by applying what we call *size-doubling techniques* (see below).

3 Modular arithmetics on CCPs

Modular multiplication, the computation[1] of $Z = XY\,[N]$ for given k-bit numbers X, Y and N, is certainly the most useful operation in public key cryptog-

[1] we denote by $X\,[N]$ the residue of X modulo N seen as an element of $\mathbb{N}$, so that the equality $X\,[N] = Y$ be properly defined.

raphy, and its efficiency in terms of hardware throughput remains highly critical for modular exponentiation-based schemes [13, 5]. There exist many different ways of performing a modular multiplication, among which Montgomery [10], De Waleffe and Quisquater [2], Barrett [1] or Sedlak [15] provide techniques that are still the most commonly used in hardware implementations. Naturally, the internal architecture of an arithmetic coprocessor relies strongly on the choice of the multiplication technique (see [8, 12] for details).

4 Size-Doubling Techniques : Algorithmic Tools for Emulating a $2k$-bit CCP from a k-bit One

There exist some tricky algorithmic infrastructures that allow one to virtually increase (typically double) the size of the crypto-coprocessor being used. Basically, these techniques allow the execution of a 1024-bit (resp. 2048-bit) exponentiation on a 512-bit (resp. 1024-bit) processor. They consist in an additional API layer which implements an up-to-$2k$-bit arithmetic engine out of an k-bit one, thereby providing up-to-$2k$-bit modular addition, multiplication, and exponentiation. Applied to a 1024-bit processor, this would allow 2048-bit DSA, 2048-bit Diffie-Hellman Key Exchange, 2048-bit ESIGN, 2048-bit RSA verification, 4096-bit RSA signature generation, and so on.

A particularly interesting case is the instanciation of a fast 2048-bit RSA on a 1024-bit chip ($k = 1024$). As only a few manufacturers have planned 2048-bit plateforms so far, this seems to be the most useful context in which to apply such size-doubling techniques. These arithmetic algorithms rely on several simultaneous computation strategies including :

1. modular representation of long operands,
2. low-storage Chinese remaindering combinations,
3. new parallelizable Montgomery-like operations, and
4. fast modular computation techniques.

These methods make previously unavailable operations on large numbers (2048-bit regular multiplication, 2048-bit addition, 2048-bit modular addition, multiplication and squaring) available and at the disposal of public key cryptographic algorithms so that they can be used with a larger key size.

4.1 Modular Representations

Long operands (ranging from 1025 to 4096 bits) are usually given under a radix form in base 2^{1024}. To be computationally exploitable, they are at first splitted into (at most) four workable 1024-bit words

$$X \rightarrow \langle X \rangle = (X\,[a_1], X\,[a_2], X\,[a_3], X\,[a_4])\ ,$$

where a_1, a_2, a_3 and a_4 are relatively prime easy-to-generate 1024-bit constants. Clearly the representation conversion $X \rightarrow \langle X \rangle$ consists solely of four modular

reductions with a 1024-bit modulus, which is usually already implemented and directly available from the underlying arithmetic engine. Larger data appears to be much easier to handle under such a form, as computing

$$\langle X \rangle,\ \langle Y \rangle \rightarrow \langle X + Y \rangle\ , \tag{1}$$

or

$$\langle X \rangle,\ \langle Y \rangle \rightarrow \langle XY \rangle\ , \tag{2}$$

is carry-free and can be performed by independent parallel computations. It is worthwhile noticing that multiplication (2) presents a lower time complexity than would a 2k-bit classical multiplication since:

$$\sum (\log a_i)^2 < (\log \prod a_i)^2\ . \tag{3}$$

Hence, this representation leads to time savings when compared to classical radix-form oriented manipulations. Obviously one has to be carefull that the cost of recombining the operand afterwards does not reduce the newly gained benefit too much. A set of moduli $\{a_1, a_2, a_3, a_4\}$ that offers this property is said to be *radix-compliant*.

4.2 Fast CRT Combinations

The original form of operands is re-obtained, after various calculations, by combining coordinates according to Garner's Chinese remaindering implementation

$$X\left[a_1 a_2\right] = \left(\frac{X\left[a_2\right] - X\left[a_1\right]}{a_1}\left[a_2\right]\right) a_1 + X\left[a_1\right]\ , \tag{4}$$

which can be cascaded to obtain a modular-to-radix representation conversion

$$\langle X \rangle = (X\left[a_1\right], X\left[a_2\right], X\left[a_3\right], X\left[a_4\right]) \rightarrow X\left[a_1 a_2 a_3 a_4\right] = X\ .$$

The use of a radix-compliant set of moduli allows the recombination of the correct operands at almost no extra cost (only linear operations), thus leading to high-speed representation conversions.

4.3 Modular Montgomery Reduction

Once two operands $\langle X \rangle$ and $\langle Y \rangle$ are given under their modular representation, one will typically wish to compute $\langle XY\left[N\right] \rangle$ where N may be up-to-2048-bit long. For that purpose, a newly discovered efficient algorithm for computing

$$\langle X \rangle,\ \langle Y \rangle,\ \langle N \rangle \rightarrow \langle XYK\left[N\right] \rangle\ , \tag{5}$$

where K is some modulus-dependent constant, can be implemented (contact the authors for further information). Routines for 2048-bit (both modular and non modular) multiplication and 4096-bit modular reduction can be directly obtained from this procedure.

4.4 Constructing a 2048-bit Exponentiation

Classically, a Square-and-Multiply implementation will call the 2048-bit modular multiplication routine that we previously discussed while bit-scanning the exponent. As a consequence, the whole algorithm remains fully compatible with exponent-relative precomputations (signed-digit, redundant representations, $d + k\phi(N)$, and so forth).

As far as we know, only this size-doubling technique makes it possible to execute on board a 1024-bit smart card any personalized version of RSA with keysize ranging continuously from 1025 to 2048 bits, and while using any public exponent. The required key material (N, e, constants) is proportional in length to the chosen RSA-size and Table 3 gives an illustration of the memory requirements for implementing a k-bit version of RSA ($1025 \leq k \leq 2048$).

EEPROM (key size)	$0.875 \times k$ bytes
Required RAM	$0.875 \times k$ bytes

Table 1. Key Size In Size-Doubling Techniques

5 Conclusion

In this brief article we have presented the data obtained during a study of the current state of the art public-key implementations on smart cards. We have also added some information on the performance of popular secret key algorithms and hash functions such as DES, SHA-1 and MD5. Although most of the performance improvements are still made for public key implementations, some chips already feature dedicated secret key coprocessors. One question for the future is whether it will be possible to embed both on the same chip, and/or whether 16-bit and 32-bit processors (like the CASCADE chip [3]) will spark a revolution in the developing smart card market.

6 Acknowledgements and Contacts

We would like to give special thanks to all interested parties who kindly helped us to gather the technical details for the survey, particularly those that could not be found in the usual documentation. Hereafter are listed some contact names and phone numbers that the reader might find useful.

Company	Contact Name	Phone/Fax No	Email Adress
Hitachi	Nicolas Prawitz	+33 1 3463 0500 +33 1 3463 3431	Nicolas.Prawitz@hitachi-eu.com
NEC	Nicolas Bérard	+33 1 3067 5800 +33 1 3067 5899	BerardN @ef.nec.de
Philips GmbH	Stefan Philipp	+49 405613 2747 +49 405613 3045	Stefan.Philipp@hamburg.sc.philips.com
SGS-Thomson	Frédéric Barbato	+33 1 47407575 +33 1 47407910	Frederic.Barbato@st.com
Siemens	Laurent Deloo	+33 1 4922 3100 +33 1 4922 3413	Laurent.Deloo@pl.siemens.fr
Atmel	Benoit Makowka	+33 4 4253 6129 +33 4 4253 6001	bmakowka@atmel.com

References

1. P. Barrett, *Implementing the Rivest, Shamir and Adleman Public-Key Encryption Algorithm on a Standard Digital Signal Processor*, Advances in Cryptology : Crypto'86 (A. M. Odlyzko ed.), LCNS 263, Springer-Verlag, pp. 311-323, 1987.
2. D. de Waleffe and J.J. Quisquater, *CORSAIR, A Smart Card for Public-Key Cryptosystems*, Advances in Cryptology : Crypto'90 (A. Menezes and S. Vanstone ed.), LNCS 537, Springer-Verlag, pp. 503-513, 1990.
3. J. F. Dhem, *Design of an Efficient Public-Key Cryptographic Library for RISC-based Smart Cards*, PhD Thesis, UCL, 1998.
4. W. Diffie and M. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, vol IT-22, n 6, pp. 644-654, november 1976.
5. T. El-Gamal, *A Public-Key Cryptosystem and a Signature Scheme based on Discrete Logarithms*, IEEE TIT, vol. IT-31:4, pp 469-472, 1985.
6. A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to Identification and Signature Problems*, Advances in Cryptology : Crypto'86 (A. M. Odlyzko ed.), LCNS 263, Springer-Verlag, pp. 181-187, 1987.
7. FIPS PUB 186, February 1, 1993, *Digital Signature Standard*.
8. D. Knuth, *The Art of Computer Programming*, vol. 2, Seminumerical Algorithms, pp. 248-250, 1969.
9. V. S. Miller, *Use of Elliptic Curves in Cryptography*, Advances in Cryptology: Crypto'85, Springer Verlag, pp. 417–426, 1986.
10. P. Montgomery, *Modular Multiplication without Trial Division*, Mathematics of Computations, vol. 44(170), pp. 519-521, 1985.
11. D. Naccache and D. MRaïhi *Arithmetic Coprocessors for Public-Key Cryptography : The State of the Art*, IEEE Micro, pp. 14-24, June 1996.
12. J. Omura, *A Public Key Cell Design for Smart Card Chips*, IT Workshop, Hawaii, USA, November 27-30, pp. 983-985.
13. R. Rivest, A. Shamir and L. Adleman, *A Method for Obtaining Digital Signatures and Public- Key Cryptosystems*. Communications of the ACM, vol. 21, n 2, p.120-126, February 1978.
14. C. Schnorr, *Efficient Identification and Signatures for Smart-Cards*, Advances in Cryptology: Eurocrypt'89 (G. Brassard ed.), LNCS 435, Springer-Verlag, pp. 239-252, 1990.
15. H. Sedlak, *The RSA Cryptographic Processor : The First High Speed One-Chip Solution*, Advances in Cryptology : Eurocrypt'87 (C. Pomerance ed.), LNCS 293, Springer-Verlag, pp. 95-105, 1988.

A Technical Carateristics of CCPs

Note that we call maximum modulus size (Max $|N|$) the size of the underlying architecture; algorithms using bigger moduli are achieved on some chips, but need additional software implementations. In Table 2, stars (*) indicate that the product is expected in the forthcoming months. In Table 5, they indicate actual measurements by Gemplus. Given figures for Atmel's MSC1114 are target ones whilst those of MSC0501 are actual. All RSA verifications are measured with $e = \mathrm{F}_4$. In all the DSA entries, the subgroup size is standard, that is, set to 160 bits.

Chip Name	Manufacturer	μ-P core	CCP name	Max $\lvert N \rvert$	RAM	ROM
H8/3111	Hitachi	H8/300	CoProcessor	576 bits	800 B	14 KB
H8/3112	Hitachi	H8/300	CoProcessor	576 bits	1312 B	24 KB
H8/3113*	Hitachi	H8/300	CoProcessor	1024 bits	1.5 KB	32 KB
T6N29	Toshiba	Z80	1024B	1024 bits	512 B	20 KB
T6N37*	Toshiba	Z80	1024B	1024 bits	512 B	20 KB
T6N39*	Toshiba	Z80	1024B	1024 bits	512 B	20 KB
T6N42*	Toshiba	Z80	2048B	2048 bits	512 B	20 KB
SLE44CR80S	Siemens	80C51	CCP	540 bits	256 B	17 KB
SLE66CX160S	Siemens	80C51	ACE	1100 bits	1280 B	32 KB
μPD789828*	NEC	78K0S	SuperMAP	2048 bits	1 KB	24 KB
ST16CF54B	SGS-Thomson	8-bit MCU	MAP	512 bits	512 B	16 KB
ST19CF68	SGS-Thomson	8-bit CPU	MAP	512 bits	960 B	23 KB
ST19KF16	SGS-Thomson	8-bit CPU	MAP	1088 bits	2 KB	32 KB
P83W854	Philips	80C51	FameX	2048 bits	800 B	20 KB
P83W858	Philips	80C51	FameX	2048 bits	800 B	20 KB
P83W8516	Philips	80C51	FameX	2048 bits	2304 B	32 KB
P83W8532	Philips	80C51	FameX	2048 bits	2304 B	32 KB
SmartXA	Philips	16 bits CPU	FameX	2048 bits	1.5/2 KB	32 KB
MSC0501	Atmel	68HC05	MEU	2048 bits	896 B	20 KB
MSC1114	Atmel	68HC05	MEU	2048 bits	1664 B	32 KB
AT90SCC	Atmel	AVR Risc	SC16	1024 bits	1/2.5 KB	-

Table 2. Technical Caracteristics of CCPs.

Chip Name	H8/3111-3112	H8/3113	ST16CF54B	ST19CF68	ST19KF16	MSC0501
Internal Clock	3.57 MHz	14 MHz	5 MHz	10 MHz	10 MHz	5 MHz
DES 64 bits	-	-	10 ms	-	-	4.5 ms
SHA 512 bits	-	-	15.2 ms	8.2 ms	8.2 ms	17 ms
MD5 512 bits	-	-	12 ms*	-	-	-
RSA 512 Sign CRT	202 ms	-	142 ms	70 ms	20 ms	86 ms
RSA 512 Sign	514 ms	68 ms	389 ms	195 ms	55 ms	-
RSA 512 Verify	-	-	9 ms	4.5 ms	2 ms	14 ms
RSA 768 Sign CRT	-	-	377 ms	189 ms	50 ms	222 ms
RSA 768 Sign	-	210 ms	-	-	165 ms	621 ms
RSA 768 Verify	-	-	190 ms	100 ms	3 ms	28 ms
RSA 1024 Sign CRT	-	-	800 ms	400 ms	110 ms	460 ms
RSA 1024 Sign	-	480 ms	-	-	380 ms	1416 ms
RSA 1024 Verify	-	-	265 ms	150 ms	5 ms	47 ms
RSA 2048 Sign CRT	-	-	-	-	780 ms	-
RSA 2048 Verify	-	-	-	-	100 ms	-
DSA 512 Sign	-	-	163 ms	84 ms	25 ms	-
DSA 512 Verify	-	-	283 ms	146 ms	40 ms	-
DSA 768 Sign	-	-	-	-	50 ms	-
DSA 768 Verify	-	-	-	-	80 ms	-
DSA 1024 Sign	-	-	-	-	100 ms	-
DSA 1024 Verify	-	-	-	-	160 ms	-
ECDSA 255 Sign	-	-	-	-	555 ms	-

Table 3. Public-Key Timings.

Name	EEPROM	Voltage	Ext Clock	Int Clock	Techno.	Sensors
H8/3111	8 KB	3V/5V	10 MHz	10 MHz	0.8 μm	LV, LF
H8/3112	8 KB	3V/5V	10 MHz	10 MHz	0.8 μm	LV,LF,HF
H8/3113*	16 KB	3V/5V	10 MHz	14.32 MHz	0.5 μm	LV,HV,LF,HF
T6N29	8 KB	3V/5V	-	-	0.6 μm	V
T6N37*	8 KB	3V/5V	-	-	-	V/T/F
T6N39*	8 KB	3V/5V	-	-	-	V/T/F
T6N42*	8 KB	3V/5V	-	-	-	V/T/F
SLE44CR80S	8 KB	3V to 5V	7.5 MHz	7.5 MHz	0.7 μm	-
SLE66CX160S	16 KB	2.7V to 5.5V	7.5 MHz	7.5 MHz	0.6 μm	-
μPD789828*	8 KB	1.8V to 5.5V	5 MHz	40 MHz	0.35 μm	-
ST16CF54B	4 KB	5V $\pm$ 10%	5 MHz	5 MHz	-	-
ST19CF68	8 KB	3V/5V $\pm$ 10%	10 MHz	10 MHz	0.6 μm	-
ST19KF16	16 KB	3V/5V $\pm$ 10%	10 MHz	10 MHz	0.6 μm	-
P83W854	4 KB	2.7V to 5.5V	8 MHz	-	-	V/T/F
P83W858	8 KB	2.7V to 5.5V	8 MHz	-	-	V/T/F
P83W8516	16 KB	2.7V to 5.5V	8 MHz	-	-	V/T/F
P83W8532	32 KB	2.7V to 5.5V	8 MHz	-	-	V/T/F
MSC0501	4 KB	2.7V to 5.5V	5 MHz	5 MHz	0.8 μm	-
MSC1114	32 KB	2.7V to 5.5V	20 MHz	10 MHz	0.4 μm	-
AT90SCC	8/48 KB	3V to 5V	-	16 MHz	0.5 μm	V/F

Table 4. Technical Caracteristics of CCPs (continued).

Chip Name	P83W854/8	P83W8516/32	SLE44CR80S	SLE66CX160S	PD789828	MSC1114
Internal Clock	-	-	5 MHz	5 MHz	40 MHz	10 MHz
DES 64 bits	10 ms	10 ms	3.7 ms*	3.7 ms*	4 ms	2.2 ms
SHA 512 bits	10 ms	5 ms	5.6 ms*	5.6 ms*	$\leq$ 2 ms	8 ms
MD5 512 bits	-	-	9 ms*	9 ms*	-	-
RSA 512 Sign CRT	45 ms	37 ms	60 ms	37 ms	16 ms	34 ms
RSA 512 Sign	140 ms	93 ms	220 ms	110 ms	52 ms	-
RSA 512 Verify	22 ms	10 ms	20 ms*	10.3 ms*	2 ms	5 ms
RSA 768 Sign CRT	182.5 ms	88 ms	250 ms*	124 ms*	52 ms	80 ms
RSA 768 Sign	385 ms	220 ms	-	437 ms*	164 ms	210 ms
RSA 768 Verify	36 ms	18 ms	-	18.4 ms*	4 ms	9 ms
RSA 1024 Sign CRT	250 ms	160 ms	450 ms	230 ms	100 ms	165 ms
RSA 1024 Sign	800 ms	400 ms	-	880 ms	360 ms	480 ms
RSA 1024 Verify	50 ms	25 ms	-	24 ms*	7 ms	16 ms
RSA 2048 Sign CRT	2180 ms	1100 ms	-	1475 ms*	750 ms	-
RSA 2048 Sign	21 s	6.4 s	-	44 s*	-	-
RSA 2048 Verify	156 ms	54 ms	-	268 ms*	45 ms	-
DSA 512 Sign	75 ms	58 ms	95 ms	50 ms	31 ms	-
DSA 512 Verify	115 ms	82 ms	175 ms	90 ms	70 ms	-
DSA 768 Sign	145 ms	100 ms	-	-	57 ms	-
DSA 768 Verify	230 ms	145 ms	-	-	150 ms	-
DSA 1024 Sign	215 ms	150 ms	-	143 ms*	-	-
DSA 1024 Verify	355 ms	225 ms	-	271 ms*	-	-
ECDSA 135 Sign	-	-	185 ms	185 ms	-	-
ECDSA 135 Verify	-	-	360 ms	360 ms	-	-
ECDSA 255 Sign	-	-	-	-	81 ms	-
ECDSA 255 Verify	-	-	-	-	380 ms	-

Table 5. Public-Key Timings (continued).