Hardware Implementation of the GPS authentication

Mickaël Dardaillon, Cédric Lauradoux, and Tanguy Risset
Université de Lyon, INRIA,
INSA-Lyon, CITI,
F-69621, Villeurbanne, France

***Abstract*—GPS is a reference in the domain of lightweight cryptography. This public-key authentication protocol was selected in the NESSIE competition and is mentioned in the standard ISO/IEC 9798. Different trade-offs between the area and the speed are explored in this paper. The fastest implementation is 40 times faster than the smallest but it is 5 times larger in term of area. At the prover side, the authentication can be done in 1μs.**

***Index Terms*—GPS, parallel/serial implementation, multiplication by a constant.**

I. Introduction

Cryptographic primitives like block ciphers or hash functions are often designed to satisfy two opposite constraints of embedded systems: speed and area. The Girault, Poupard and Stern authentication protocol (GPS) is a well-established primitive in the cryptographic community: it has been proven secure (NESSIE recommendation [1]) and it can be implemented with a low budget. GPS is a reference in the domain of lightweight authentication. However, it is unknown yet how fast can be GPS. This question finds its answer here.

The core of GPS consists in implementing a *multiplication by a constant* operator. Two strategies can be adopted to implement this arithmetic unit: (a) a parallel approach based on *partial pre-computation*, and (b) a serial approach based on *shift-and-add*. The bounds for GPS are established in terms of area and speed for these two approaches.

The parallel implementation of GPS with a 128-bit secret takes 1μs against 42μs for the existing serial implementation on Xilinx Virtex 4 at 8 Mhz. If the parallel approach is affordable for a powerful FPGA (Virtex 4), lower resource platforms are not yet ready to support such implementation. Our full integration of GPS in the communication stack of the PowWow platform support only the serial implementation. For such a platform to benefit from the fastest GPS implementation, twice more area would be needed for 128-bit secret.

The rest of this paper is organized as follows. The GPS protocol is reminded in Section II with its existing implementation. The Section III and IV describes the different trade-off available for the implementation of GPS. The performances are compared in Section V.

II. GPS protocol

The GPS authentication protocol [1] is an interactive zero-knowledge authentication protocol initially proposed by Girault, Poupard, and Stern [2]. It provides provable security based on the composite discrete logarithm problem. It also combines short transmissions and minimal on-line computation, using precomputed "coupons". This protocol has been selected in the NESSIE portfolio [3] and it is mentioned in the ISO/IEC 9798-5 Clause 8 [4] as a reference. Throughout the paper, we implicitly refer GPS as this variant "with coupons".

a) Description: The parameters used in this protocol are the following:

- S, C, D are public integers, where $|S| \approx 180$, $|C| = 32$ and $|D| = |S| + |C| + 80$,
- $n = p \times q$ is a public composite modulus, where p and q are secret primes, $|n| = 1024$, $|p| = |q| = 512$,
- g is an element of $\mathbb{Z}_n^*$,
- $\Phi = (C-1) \times (S-1)$,
- $s \in [0, S[$ and $I = g^{-s} \mod n$,
- a coupon i is a couple $(r_i, x_i = g^{r_i} \mod n)$, where $r_i \in [0, D[$ is a random number.

At the beginning, the prover P has a unique identifier Id_P, a unique pair of keys (the private s and the public I) and a set of coupons c computed by a higher trusted entity. The verifier V knows the prover's identifier and public key.

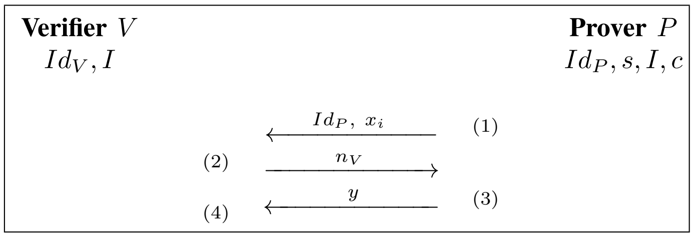

Fig. 1. The GPS protocol.

GPS, depicted in Fig. 1, works as follows.

(1) The prover P chooses a coupon (r_i, x_i), and sends its identifier Id_P and x_i to the verifier R.
(2) The verifier answers a challenge n_V randomly chosen in the interval $[0, C[$.
(3) The prover computes $y = r_i + n_V \times s$, and sends y to the verifier.
(4) The verifier checks if:
    - $g^y \times I^{n_V} \mod n = x_i$
    - $y \in [0, D + \Phi[$

The arguments supporting the security of GPS can be found in [1], [2]. They are not included because the security of GPS is not affected by our results.

b) Existing implementation: GPS authentication has been designed for constraint embedded systems such as smartcard, RFID or sensors networks. The critical part for the implementation is on the prover side assuming that the verifier (RFID reader or a base station) suffers from less restriction.

Two steps are critical for the prover: the computation of x_i (Step (1)) and y (Step (3)). The computation of x_i is the most expensive one (exponentiation) but the prover has nothing to do thanks to the coupons (pre-computation). Therefore, the last remaining cost is the Step (3) which consists of the multiplication by s and the addition of r_i. The multiplication by s is the most complex due to the size of the multiplicands.

GPS was first designed for smartcards [2]. McLoone and Robshaw proposed in [5], [6] the first hardware implementation of GPS. Their solution is based on the shift-and-add algorithm for multiplication. It offers a very small hardware footprint. This implementation is described in Section IV and it serves as the reference in our comparison.

Girault and Lefranc [7] proposed a variant of GPS which exploits low Hamming weight secret s. The multiplication is transformed in an addition when the secret is chosen properly. This variant can significantly reduced the cost of GPS. However, this solution was subsequently attacked in [8], and it is not considered in this work.

In the following sections, the different architectures for GPS are explored. Two criteria are examined: the *area* and the *speed*. Area and speed are two classical metrics to compare hardware designs. A cryptographic design can be tuned for a specific key or support any value for the key. Throughout this paper, fixed-key and variable-key implementation are considered.

III. Parallel implementation

A parallel multiplier can be implemented using lookup table (ROM). Let us consider two variable operands A and B, of size $|A|$ and $|B|$. The product $A \times B$ is on $|A| + |B|$ bits. A lookup table approach requires to store: $2^{|A|+|B|} \times (|A| + |B|)$ bits. While being attractive for small values of A and B, lookup tables are clearly not practicable for GPS, *i.e.* $|A| \geq 32$. This cost can be reduced to $2^{|A|} \times (|A| + |B|)$ considering that B is fixed, *i.e.* the key is fixed.

To further reduces the memory size, Chapman proposed in [9] to decompose the computation into partial products. To understand the principle of this decomposition, basic multiplication method from elementary school is reminded in Fig. 2.

$$\begin{array}{llcr}
 & & & 482 \\
 & & \times & 7 \\ \cline{3-4}
2 \times 7 & \rightarrow & & 14 \\
8 \times 7 \times 10 & \rightarrow & + & 560 \\
4 \times 7 \times 100 & \rightarrow & + & 2800 \\ \cline{3-4}
 & & & 3374
\end{array}$$

Fig. 2. Elementary school multiplication

In order to compute multiplication efficiently, human decomposes numbers into digits in decimal basis. Then, we learn multiplication tables for all the digits. Multiplication is done as shown in Fig. 2 using these tables, left-shift ($\times 10, \times 100, \cdots$) and digits addition. If we have to compute fast multiplications involving a specific constant, we only need the multiplication table of this constant. We illustrate this on Fig. 3.

$$\begin{array}{llcr}
 & & & 953 \\
 & & \times & 482 \\ \cline{3-4}
2 \times 953 & \rightarrow & & 1906 \\
8 \times 953 \times 10 & \rightarrow & + & 76240 \\
4 \times 953 \times 100 & \rightarrow & + & 381200 \\ \cline{3-4}
 & & & 459346
\end{array}$$

Fig. 3. Constant multiplication by 953.

For an hardware design, the decomposition is done in the 2^ℓ basis rather than the decimal basis. The choice of ℓ depends on the trade-off between the cost (memory and adders) and the technology characteristics. First, 2^ℓ values need to be stored in each table. They represents $\frac{|A|}{\ell}$ tables of $2^\ell \times (|B| + \ell)$ bits. The results obtained from these tables are combined by $\frac{|A|}{\ell} - 1$ adders of $|B| + \ell$ bits. In most FPGA, the ROM are composed of four binary inputs, which leads to a size $\ell = 4$ bits. This architecture is illustrated on Fig. 4 for $|B| = 8$, $|A| = 12$ and $\ell = 4$. Each lookup tables take one part of A as an input. The partials results are combined using 2 adders to produce the final result. Note that the left shifts are done by positioning the partial results and have no gate cost.

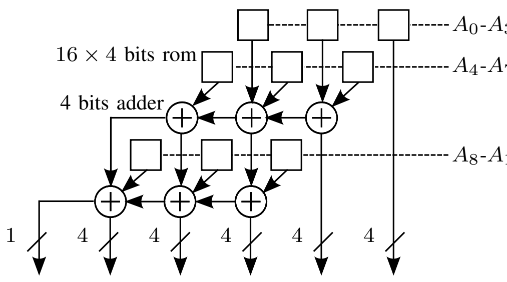

Fig. 4. Constant multiplier (adapted from [10]).

For GPS, we evaluate architectures with 128, 256 and 512 bits secrets and 32 bits challenges. The architectures are generated automatically in our system, using the library FloPoCo[1]. This library is an open source project for non-standards operations and it provides in particular several integer constant multipliers [11].

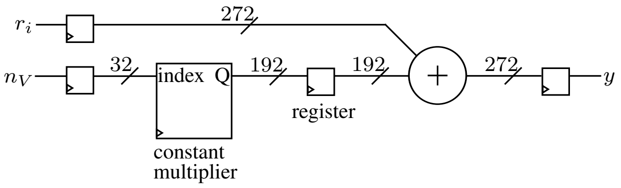

Fig. 5. Parallel implementation of GPS (128 bits).

The Fig. 5 illustrates the parallel implementation of GPS used in our design. The product is computed by the constant multiplier generated core, with the private key s as a constant. The result $s \times n_V$ is added to r_i. All the data is processed in parallel.

This architecture offers the advantage of a full parallel implementation in terms of speed, while using the constant property of the key to reduce the size of the implementation.

[1]http://flopoco.gforge.inria.fr

IV. Serial implementation

The serial implementation uses the McLoone and Robshaw architecture to get very low hardware requirements [5]. This architecture is based on the shift and add algorithm to compute the product. This algorithm is similar to the elementary school multiplication. For each digit of the multiplicand, we compute the product between the digit and the second multiplicand. The result is shifted to the left accordingly to the digit position. The sum of the results is the multiplication result.

$$\begin{array}{lccr}
 & & & 101001 \\
 & & \times & 110 \\
\cline{3-4}
0 \times 101001 & \rightarrow & & 000000 \\
1 \times 101001 \times 10 & \rightarrow & + & 1010010 \\
1 \times 101001 \times 100 & \rightarrow & + & 10100100 \\
\cline{3-4}
 & & & 11110110
\end{array}$$

Fig. 6. Elementary school binary multiplication

The multiplication is illustrated on Fig. 6 for binary numbers. The product between a multiplicand and a binary digit is either 0 or the multiplicand, depending on the binary digit. Starting from the bottom of Fig. 6, each iteration reads a digit of one operand. If this digit is 1, it adds the multiplicand to the previous results, otherwise, it adds nothing (or 0). The result of this sum is shifted one bit to the left.

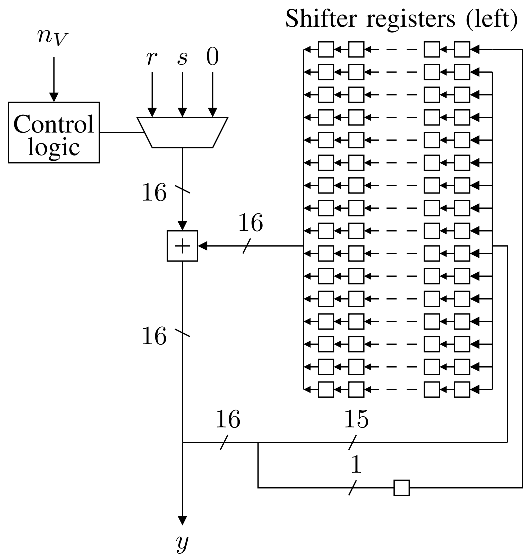

Fig. 7. GPS serial implementation (from [5]).

The architecture based on this shift and add algorithm is presented on Fig. 7. The control logic block process the challenge bitwise and drives the multiplexer. This multiplexer select the key or 0, depending on the challenge bit. The adder sums the multiplexer data with the previous results stored in the shift register. On the last turn, r_i is added to the result, using the same hardware. The data is processed by blocks to obtain a serial architecture. The balance between occupied area and execution time is set by the data bus size. Doubling the data bus size halve the required run cycles, while increasing the adder and the multiplexer size.

Our implementation uses the same model as the McLoone and Robshaw, and we get area results close to the original for the different security parameters (see the appendix). However, these results only takes into account the computing hardware. For the rest of the article, we will give results for a complete implementation. This implementation contains the McLoone and Robshaw architecture, as well as memories for n_V, x_i and y.

V. Comparison

From Section III and IV, two different approaches to implement GPS have been described. One focuses on speed at the expense of a large hardware footprint, while the other focuses on the lowest footprint. The two designs are compared and analyzed in terms of speed and area. In order to validate our implementation, we use a setup similar to the one used in [5]. The results matches as shown in the Appendix.

secret size	128 bits	256 bits	512 bits
parallel impl. slices	2213	4358	7115
serial impl. slices	493	784	1377
parallel/serial ratio	4.5	5.5	5.2
parallel impl. run cycles	8	12	20
serial impl. run cycles	339	603	1131
serial/parallel ratio	42.4	50.3	56.55

TABLE I
Comparison between parallel and serial implementation.

Table I compares the serial and the parallel implementation in terms of hardware footprint and run cycles at 8 MHz on a Xilinx Virtex 4 XC4VSX35. The comparisons are made on the overall architecture, including all the required memories. The results shows that the serial implementation is 5 times smaller than the parallel implementation. This is a significant advantage in terms of hardware footprint. On the other hand, the parallel implementation is more than 40 times faster than the serial implementation. Knowing that the constant multiplier is an off-the-shelf component, it is a very significant advantage in terms of execution speed.

To continue further our study of GPS, we work on the wireless sensor platform PowWow[2]. This platform is specifically designed to be energy efficient [12], and offers a more constraint hardware platform in terms of area and time. It uses a low power Actel Igloo AGL250 FPGA for control and a Texas Instrument CC2420 chip for RF communications.

We integrate GPS authentication into the network stack of the platform. Our demonstrator uses GPS to authenticate a PowWow platform (Prover) to a base node (Verifier). The experimental set-up is illustrated on Fig. 8.

During our experiments, we made two observations. First, the parallel implementation of GPS is too large to fit in the Igloo AGL250 FPGA. Second, the bus size impacts the serial implementation of GPS. This effect is exposed in the Table II.

From these results, it can be seen that the 16-bit data bus is offering a lower hardware footprint than the 8-bit data bus. This is due to an additional overhead needed to address the

[2]http://powwow.gforge.inria.fr

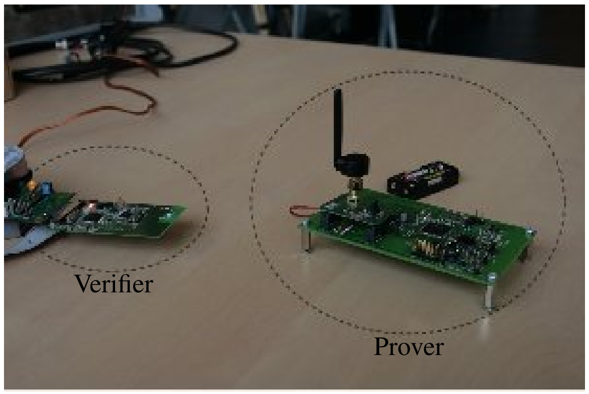

Fig. 8. PowWow running a wireless authentication.

	core cells for different secrets size		
bus width	GPS 128	GPS 256	GPS 512
8 bits	1542	2270	3745
16 bits	1546	2253	3698
32 bits	1934	2632	4034

TABLE II
INFLUENCE OF THE DATA BUS SIZE ON THE HARDWARE FOOTPRINT.

memory for the 8-bit data bus, which is not balanced by the shift and add architecture size reduction.

VI. CONCLUSION

Two architectures to implement the GPS cryptosystem have been presented. The first one is the parallel architecture based on a constant multiplier. The second one is the McLoone and Robshaw architecture [5]. Our results are conformed to the existing implementation (see the Appendix). Up to our knowledge, this is the first time that the parallel architecture is explored. Moreover, our experimentations include a full integration of GPS into the PowWow platform and validate its applicability in a realistic set-up.

Our work gives an insight on the gap between the serial and the parallel implementation in terms of area and timing. Analysing the results from our experiments, we saw that a smaller data bus does not always imply a smaller hardware footprint. A possible leverage to improve the hardware footprint of the parallel implementation could be the exploration of dedicated hardware optimisations for the constant multiplier. It will be explored in further works.

ACKNOWLEDGEMENTS

The authors wants to thank Florent de Dinechin from ENS Lyon for his help on constant multiplication and Romain Fontaine from the CAIRN INRIA team for his help on the RF-communication of PowWow platform.

REFERENCES

[1] O. Baudron, F. Boudot, P. Bourel, E. Bresson, J. Corbel, L. Frisch, H. Gilbert, M. Girault, L. Goubin, J.-F. Misarsky, P. Nguyen, J. Patarin, D. Pointcheval, G. Poupard, J. Stern, and J. Traor, "GPS - An Asymmetric Identification Scheme for on the Fly Authentication of Low Cost Smart Cards," A proposal to NESSIE, 2001, https://www.cosic.esat.kuleuven.be/nessie/updatedPhase2Specs/gps/GPS-Bv2.pdf.

[2] M. Girault, G. Poupard, and J. Stern, "On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order," *Journal of Cryptology*, vol. 19, no. 4, pp. 463–487, 2006.

[3] NESSIE consortium, "Portfolio of recommended cryptographic primitives," Tech. Rep., 2003, https://www.cosic.esat.kuleuven.be/nessie/deliverables/decision-final.pdf.

[4] International Organization for Standardization, "ISO/IEC 9798 – Information technology – Security techniques – Entity authentication," ISO, 1997 – 2008, http://www.iso.org.

[5] M. McLoone and M. J. Robshaw, "Public Key Cryptography and RFID Tags," in *The Cryptographers' Track at the RSA Conference – CT-RSA*, ser. Lecture Notes in Computer Science 4377. San Francisco, CA, USA: Springer-Verlag, February 2007, pp. 372–384.

[6] M. McLoone and M. J. B. Robshaw, "New Architectures for Low-Cost Public Key Cryptography on RFID Tags," in *International Symposium on Circuits and Systems - ISCAS 2007*. New Orleans, LO, USA: IEEE, May 2007, pp. 1827–1830.

[7] M. Girault and D. Lefranc, "Public Key Authentication with One (Online) Single Addition," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, ser. Lecture Notes in Computer Science 3156. Cambridge, MA, USA: Springer, August 2004, pp. 413–427.

[8] J.-S. Coron, D. Lefranc, and G. Poupard, "A New Baby-Step Giant-Step Algorithm and Some Applications to Cryptanalysis," in *Cryptographic Hardware and Embedded Systems - CHES 2005*, ser. Lecture Notes in Computer Science 3659. Edinburgh, UK: Springer, August 2005, pp. 47–60.

[9] K. D. Chapman, "Constant Coefficient Multipliers for the XC4000E," Tech. Rep., 1996.

[10] M. J. Wirthlin, "Constant Coefficient Multiplication Using Look-Up Tables," *The Journal of VLSI Signal Processing*, vol. 36, no. 1, pp. 7–15, Jan. 2004.

[11] N. Brisebarre, F. de Dinechin, and J.-M. Muller, "Integer and floating-point constant multipliers for FPGAs." Leuven, Belgium: IEEE Computer Society, July 2008, pp. 239–244.

[12] O. Berder and O. Sentieys, "PowWow : Power Optimized Hardware/Software Framework for Wireless Motes," in *International Conference on Architecture of Computing Systems - ARCS '10*. Hannover, Germany: VDE Verlag, February 2010, pp. 229–234.

APPENDIX

Table III compares the results obtained from our serial implementation with the result obtained in [5]. The synthesis was done on the Cadence ATL Compiler and normalized to a NAND size. Our results are less than 10 % different from the article, which means that our implementation is close to the article, and can be used as a benchmark.

Key	Challenge	Area (NAND)		Difference
(bits)	(bits)	[5]	this letter	
8 bits adders				
160	32	1541	1505	2.31%
128	32	1327	1320	0,54%
160	20	1486	1413	4,89%
128	20	1286	1231	4,28%
160	8	1371	1341	2,21%
128	8	1167	1163	0,37%
16 bits adders				
160	32	1642	1594	2,90%
128	32	1411	1403	0,54%
160	20	1642	1502	8,53%
128	20	1411	1314	6,90%
160	8	1511	1395	7,65%
128	8	1298	1205	7,16%

TABLE III
PERFORMANCES COMPARISON BETWEEN THIS LETTER AND [5].