Hardware Implementation of the GPS authentication

Mickaël Dardaillon, Cédric Lauradoux, Tanguy Risset

Hardware Implementation of the GPS authentication

Mickaël Dardaillon, Cédric Lauradoux and Tanguy Risset

Université de Lyon, INRIA,

INSA-Lyon, CITI-INRIA, F-69621, Villeurbanne, France

Emails: {mickael.dardaillon, tanguy.risset}@insa-lyon.fr, cedric.lauradoux@inria.fr

Abstract—In this paper, we explore new area/throughput trade-offs for the Girault, Poupard and Stern authentication protocol (GPS). This authentication protocol was selected in the NESSIE competition and is even part of the standard ISO/IEC 9798. The originality of our work comes from the fact that we exploit a fixed key to increase the throughput. It leads us to implement GPS using the Chapman constant multiplier. This parallel implementation is 40 times faster but 10 times bigger than the reference serial one. We propose to serialize this multiplier to reduce its area at the cost of lower throughput. Our hybrid Chapman's multiplier is 8 times faster but only twice bigger than the reference. Results presented here allow designers to adapt the performance of GPS authentication to their hardware resources. The complete GPS prover side is also integrated in the network stack of the PowWow sensor which contains an Actel IGLOO AGL250 FPGA as a proof of concept.

Keywords-GPS, parallel/serial implementation, multiplication by a constant.

I. INTRODUCTION

A current trend in cryptography is the design of primitives with a low-cost implementation. *Lightweight cryptography* [1] is interested in metrics such as the circuit area, energy consumption or code size. The Girault, Poupard and Stern authentication protocol (GPS) is one of the oldest lightweight primitives which has been recommended by the European project NESSIE [2] and appeared in the ISO/IEC 9798-5 standard [3]. GPS is a reference in the domain of lightweight authentication and the existing implementations aim to reduce as much as possible the area. In this paper, we explore the trade-off between speed and area in the implementation of GPS.

Implementing GPS consists in designing an adder and a multiplier. The latter is the most critical part in a GPS core. Three typical strategies can be followed to implement this core: *serial*, *parallel* and a trade-off between the previous two that we designate by *hybrid*. The serial implementation based on *shift-and-add* has been already covered in [4] and is the baseline for area. The other two strategies have not yet been studied. The parallel implementation is the traditional method to achieve the best speed at the drawback of a large area. The hybrid implementation in our paper is in fact a serialization of the parallel approach that takes advantage of hardware repetition. It provides a trade off between serial/parallel approaches by executing chunks of parallel multiplication.

To implement a parallel implementation of GPS, we face a big challenge: implementing parallel multiplier with large variable operands is out-of-question. To overcome this limitation, we have specialized our parallel and hybrid implementations for a specific key. The complexity of our multiplier is therefore reduced to the implementation of a *multiplier by a constant* [5].

We attempt to integrate our different implementations in the network stack of the wireless sensor platform PowWow. This platform embeds an Actel IGLOO FPGA with a 8 MHz clock cycle. Our simulation results for a security level of 128-bit show that the parallel implementation of GPS with a secret takes $1\mu s$ against $42\mu s$ for the existing serial implementation at 8 MHz. However, it does not fit within the FPGA. The integration of GPS into our platform is only possible for hybrid and serial implementation.

We note that the basic GPS protocol requires a $320\mu s$ response time, but McLoone and Robshaw circumvent this difficulty by using a different protocol in their article [4] with a $18ms$ latency. Our implementation fulfil the $320\mu s$ constraint for all architectures at 8 MHz, and would help reach this goal at a lower frequency for parallel and hybrid implementations.

The rest of this paper is organized as follows. The GPS protocol is reminded in Section II. The Section III describes the existing serial implementation of GPS. The main contributions of the paper are given in Section IV and V with the description of the parallel and hybrid implementation. The performances are compared and analyzed in Section VII.

II. GPS PROTOCOL

The GPS authentication protocol [2] is an interactive zero-knowledge authentication protocol initially proposed by Girault, Poupard, and Stern [6]. 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 [7] and it is mentioned in the ISO/IEC 9798-5 Clause 8 [3] 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 $|S| \approx 180$ bits[1], $|C| = 32$ and $|D| = |S| + |C| + 80$,
- $n = p \times q$ is a public composite modulus, where p and q are secret primes, $|n| = 1024$, $|p| = |q| = 512$,

---

[1] In all the paper we use the notation $|X|$ for *the size in bits of number X*

- 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.
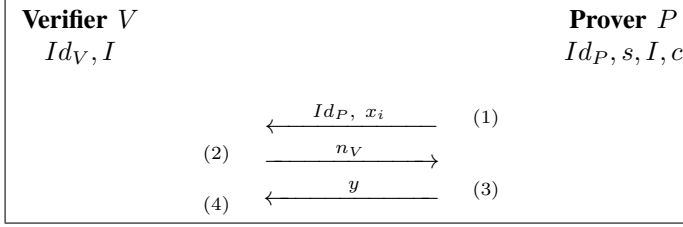

Fig. 1. The GPS protocol.

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 [2], [6]. They are not included because the security of GPS is not affected by our results.

In the remaining of the paper, s is named *the secret*, n_V is named *the challenge* and r_i *the commitment* (see [2]).

b) Existing implementation: GPS authentication has been designed for constraint embedded systems such as smart cards, 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 smart cards [6]. McLoone and Robshaw proposed in [4], [8] 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 III and it serves as the reference in our comparison.

Girault and Lefranc [9] 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 reduce the cost of GPS. However,

this solution was subsequently attacked in [10], and it is not considered in this work.

In the following sections, the different architectures for GPS are explored. Two criteria are examined: *area* and *speed*. 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. SERIAL IMPLEMENTATION

Fig. 2. Shift-and-add classical binary multiplication.

McLoone and Robshaw have proposed in [4] the reference serial implementation of GPS. This architecture is based on the classical shift-and-add multiplication (Fig. 2). The core of the design is a 16-bit adder. The product $n_V \times s$ is decomposed into a succession of 16-bit additions. The same 16-bit adder is re-used to perform the final addition $n_V \times s + r_i$ in 16-bit chunks.
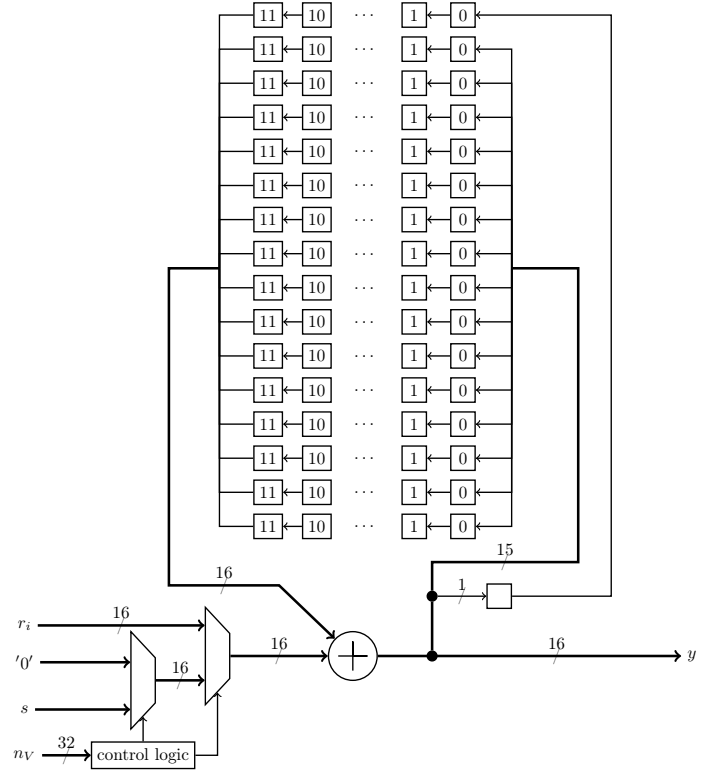

Fig. 3. GPS serial implementation for a 128-bit secret and a 16-bit adder (from [4]).

The architecture of McLoone and Robshaw is presented on Fig. 3. The control logic block process the challenge n_V bitwise and drives the multiplexers. The first multiplexer selects 16 bits of the secret s or 0, depending on the challenge

bit. The adder sums the multiplexer data with the previous results stored in the shift register. The final addition of r_i with the result is controlled by the second multiplexer. The same hardware is re-used for this addition. The data is processed sequentially by 16-bit 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.

We have implemented the serial architecture of McLoone and Robshaw independently, and we get area results close to their original article for the different security parameters (see Table I). However, McLoone and Robshaw results only take into account the computing hardware. We give results for a complete implementation of the serial multiplier, including memories for n_V, r_i, s and y.

IV. PARALLEL IMPLEMENTATION

A multiplier can be implemented using lookup tables (ROM). Let us consider two variable operands n_V and s, of size $|n_V|$ and $|s|$. The product $n_V \times s$ has a $|n_V| + |s|$ bits width. A lookup table approach requires to store:

$$2^{|n_V|+|s|} \times (|n_V| + |s|) \text{ bits.}$$

While being attractive for small values of n_V and s, lookup tables are clearly not practicable for GPS, *i.e.* $|n_V| \geq 32$.

A first attempt to reduce this cost consists in fixing s. This hypothesis implies that our implementation is dedicated to a given key. In most cryptographic implementations, the key can be updated. To our knowledge, there are two exceptions. FPGA supplier provides encryption scheme with fixed key to protect the users bit-stream. White-box cryptography [11] also uses implementation with fixed key but to obfuscate the key into the code. Our goal is to show that implementation with fixed key can provide benefits in term of throughput.

Using this assumption, the cost of a table lookup multiplier can be reduced to:

$$2^{|n_V|} \times (|n_V| + |s|) \text{ bits.}$$

This approach is still not reasonable for GPS.

To further reduce the memory size, Chapman proposed in [12] the KCM method (for *k constant multiplier*) to decompose the computation into partial products. Let us explain Chapman's KCM method in decimal basis (Fig. 4) for the multiplication of $953 \times x$. If one can store in look-up tables (LUT) the nines values 1×953, 2×953,..., 9×953, then one can compute the whole product by simply adding stored values as indicated in the Fig. 4 for 953×482.

$$
\begin{array}{r}
953 \\
\times \quad 482 \\
\hline
\end{array}
$$

$4 \times 953 \times 100$	$\rightarrow$	$+ \quad 3812..$
$8 \times 953 \times 10$	$\rightarrow$	$+ \quad 7624.$
2×953	$\rightarrow$	1906
		459346

Fig. 4. Chapman KCM principle for multiplication by 953 in decimal basis.

For an hardware design, KCM uses 2^ℓ basis rather than the decimal basis. The choice of ℓ depends on the trade-off between the cost (memory and adder) and the technology characteristics (number of inputs per lookup table). For example, a classic Xilinx Virtex 4 as 4-input LUT, and our Actel Igloo as 3-input LUT. First, 2^ℓ values need to be stored in each table, hence for a total memory size of:

$$\frac{|n_V|}{\ell} \times 2^\ell \times (|s| + \ell) \text{ bits.}$$

The results obtained from these tables are combined by $\frac{|n_V|}{\ell} - 1$ adders of $|s| + \ell$ bits.
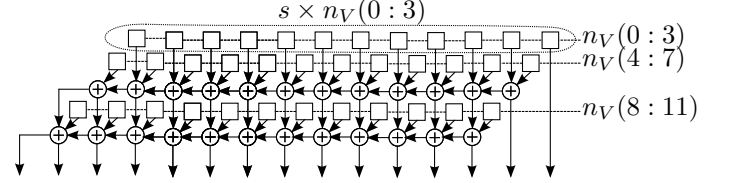

Fig. 5. KCM constant multiplier (adapted from [5]).

This architecture is illustrated on Fig. 5 for $|s| = 8$, $|n_V| = 12$ and $\ell = 4$. Each lookup table takes one part of n_V 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. Note also that the result is obtained in a single cycle, with a very long critical path that could slow down the clock cycle of the circuit. To cope with this clock cycle problem, the designer can pipeline the circuit, increasing latency but maintaining a throughput of 1 result per clock cycle.

For our GPS implementation of the parallel approach, we used a KCM operator generated by the FloPoCo library[2]. This library is an open source project for non-standards arithmetic operators. It provides in particular several integer constant multipliers [13]. All operators are pipelined to run at high frequency (default setting at 300 MHz on Virtex FPGA).
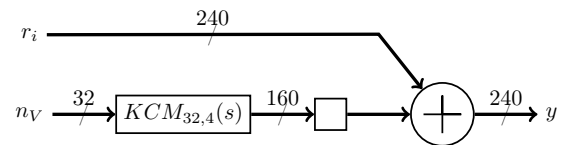

Fig. 6. Parallel implementation of GPS (128 bits).

The Fig. 6 illustrates the parallel implementation of GPS used in our design. The notation $KCM_{32,4}(s)$ stands for *Chapman constant multiplier of a 32-bit number with the constant s, using 4-input LUT*. The product is computed by the constant multiplier core generated with FloPoCo, with the secret s as a constant. The result $s \times n_V$ is added to r_i.

This architecture is the fastest in terms of speed using the property that the secret s is constant to reduce the size of the implementation. The performance results of the parallel approach are presented further in the paper, but the design could not fit on a small embedded FPGA such as the IGLOO

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

AGL250 used in PowWow, hence the need of a trade-off between the serial and parallel approach.
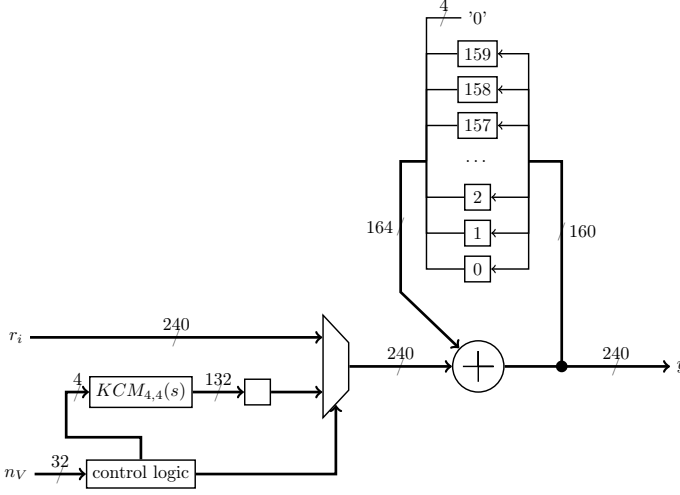
V. Hybrid implementation

Fig. 7. Hybrid KCM architecture (128 bits).

The idea of the hybrid implementation is to serialize the KCM architecture of Fig. 5 *horizontally*. Indeed, each horizontal row of LUT is the same: it contains all the values of s multiplied by $0, 1, \ldots, 2^\ell - 1$. Hence if we sequentialize the addition of one row with the others we may use a single LUT of size $2^\ell \times |s| + \ell$. The challenge n_V is sent by blocks of four bits to drive the LUT, the output of the LUT is added to the accumulation register and then shifted by four bits for the next addition.

The architecture, illustrated on Fig. 7, uses one KCM table of $2^\ell \times (|s| + \ell)$ bits to compute the product between ℓ bits of the challenge and the constant s. The result is accumulated and shifted by ℓ bits each cycle. The adder is reused to sum the result of the multiplication with r_i. All the computation is done using a parallel adder. To further reduce the critical path, the adder could also be sequentialized with a sequence of smaller adders as it was done for the serial approach in section III.

VI. Experimental platform

We performed our implementation on the wireless sensor platform PowWow[3]. This platform is specifically designed to be energy efficient [14], and offers a constraint hardware platform in terms of area and time. It uses the Actel IGLOO AGL250 FPGA for control and a Texas Instrument CC2420 chip for RF communications. Actel IGLOO uses non volatile flash technology for FPGA dedicated to low power.

We integrated the GPS authentication into the network stack of the platform. The physical layer is based on the 802.15.4 standard with a CSMA/CA access provided by the CC2420 and controlled by the FPGA. It supports a simple point to point access with the verifier in order to process the whole

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

GPS authentication (see Fig. 1 for details). In our set-up, the verifier is a computer interfaced with a Senslab[4] node for telecommunications.

The implementation was verified by authenticating the prover to the verifier 8 times in a row, and with different keys on separate synthesis, using vectors generated by GMP[5] as reference. The serial architecture was evaluated with several challenge and key sizes ($|s| = \{128, 256, 512\}, |n_V| = \{16, 20, 32\}$). The hybrid architecture could be implemented on the platform for a key size up to 256 bits. The experimental set-up is illustrated on Fig. 8.
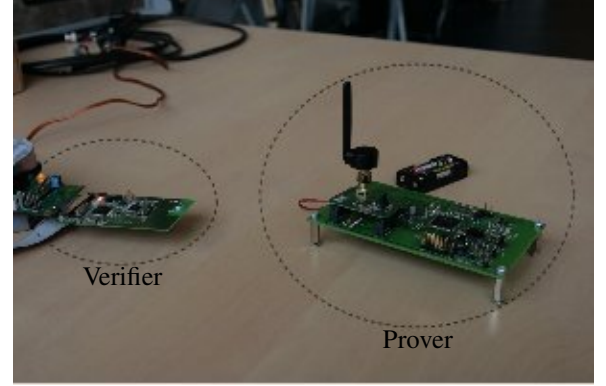

Fig. 8. PowWow running a wireless authentication.

In order to validate our work, we have implemented independently the serial approach of GPS to check that our results were compatible with those of McLoone and Robshaw in [4]. The synthesis was done on the Cadence ATL Compiler and normalized to a NAND size to be consistent with [4]. The Table I compares our area results with the existing ones. The difference is below 5 % for a serial implementation with an 8-bit adder and below 10 % with an 16-bit adder. We consider these differences as acceptable: our serial implementation can be used as a benchmark to compare the different architectures.

TABLE I
AREA COMPARISON BETWEEN OUR SERIAL IMPLEMENTATION AND THE ONE OF McLOONE AND ROBSHAW [4].

Secret (bits)	Challenge (bits)	Area (NAND) [4]	our work	Difference
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%

[4]http://www.senslab.info
[5]http://gmplib.org

VII. Implementation results

From Section III to V, three different approaches to implement GPS have been described to obtain a low area footprint, an high throughput or a trade-off between them. We now aim to verify these expectations, compare our implementations quantitatively and determine the influence of the secret size.

The three approaches discussed above were written in VHDL and different synthesis were made. All the results are given for a synthesis and a mapping performed using Actel tools and targeting Actel IGLOO AGL250. The targeted clock cycle was fixed for all impementations at 8 MHz, frequency used on the PowWow platform. During our experiments, we have aligned systematically the bus size on the adder size. Moreover, we evaluated the overall footprint, including all the required memories. This is an important point, as it turns out that memories are occupying much more space than computing parts. It also permits to illustrate the gains obtained by using a constant multiplier, which includes the constant memory.
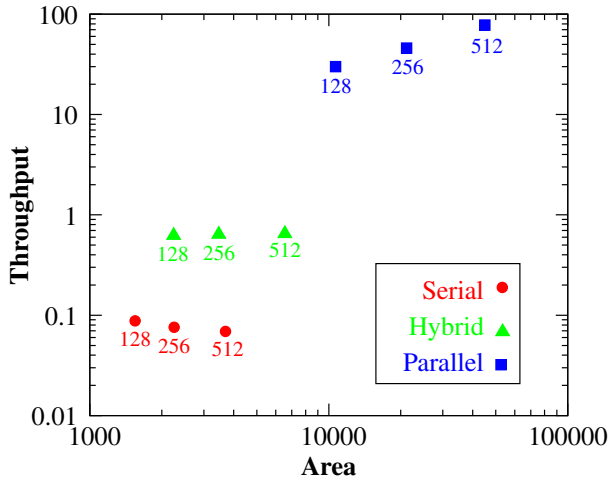

Fig. 9. Area vs throughput for a 32-bit challenge (log scale) and different secret sizes ($|s| = \{128, 256, 512\}$).

We plotted the area and throughput for the three implementations on Fig. 9. This illustrates the three different balances between area and throughput achieved by the three implementations. Moreover, these balances are maintained for the different levels of security, and both the area and throughput can be represented by a linear approximation in function of the secret size ($y = a * |s| + b$). We observe that the area is increasing with the security size for the serial ($a = 5, 6$), the parallel ($a = 89, 8$) and the hybrid ($a = 11, 3$) implementation, with the best scaling for the serial approach in regards of the area. In terms of throughput, the serial implementation as a negative slope ($a = -46, 3 * 10^{-6}$), contrary to the parallel ($a = 125 * 10^{-3}$) and hybrid ($a = 61, 9 * 10^{-6}$) implementations, which gets a better throughput by using a larger secret $|s|$.

The results shown in Table II compares the three implementations in terms of hardware area, run cycles and throughput (in cycles per byte of result). We use the serial implementation as a reference for our comparison. The parallel implementation has an area 10 times larger than the serial implementation.

TABLE II
AREA, LATENCY AND THROUGHPUT COMPARISON BETWEEN IMPLEMENTATIONS FOR A 32-BIT CHALLENGE AND DIFFERENT SECRET SIZES.

Secret (bits)	Implementation		
	Serial	Parallel	Hybrid
	Area (core cells)		
128	1546	10676	2243
256	2253	21171	3467
512	3698	44978	6553
	Latency (cycles)		
128	339	8	48
256	603	12	72
512	1131	20	120
	Throughput (cycles/byte)		
128	0,088	30	0,625
256	0,076	46	0,639
512	0,069	76	0,650

For that cost, its pipeline structure provides a new result each cycle and a 40 times smaller latency when using multi-authentication. The hybrid implementation offers a middle ground with an area less than doubled, and an 8 times smaller latency.

We also explored the impact of the adder size on the area for the serial implementation in the Table III. To reduce the area, it would seems logical to choose the smallest adder. However, it impacts also the bus size and how the memory is addressed. Therefore, choosing the smallest adder is not necessarily the best solution because it can imply an addressing overhead. The gain for the adder can be offset by the memory cost. This effect is highly dependent on the underlying technology. In our case, the 16-bit adder has a lower area than the 8-bit one.

TABLE III
IMPACT OF THE ADDER ON THE SERIAL IMPLEMENTATION AREA FOR A 32-BIT CHALLENGE.

Adder (bits)	Secret (bits)		
	128	256	512
	Area (core cells)		
8	1542	2270	3745
16	1546	2253	3698
32	1934	2632	4034

In order to be complete on the implementation footprint, we should also add the coupon footprint. By using a PRNG as suggested in the McLoone and Robshaw article [4], one can reduce the footprint to 1000 NAND equivalent for storing 20 coupons, with another 1000 NAND for the PRNG. This would require about 2300 core cells, and is compatible with the serial and hybrid implementation for 128-bit and 256-bit secret size.

VIII. Conclusion

Several architectures for GPS are presented in this work offering different balances between area and throughput. With respect to this goal, our contribution is three-fold. First, we have shown that using a fixed key can enable new implementations possibilities. Second, we have serialized KCM to reduce its area cost. To our knowledge, this is the first time that such a trade-off is proposed for KCM. Third, we have integrated our GPS cores into a real platform (PowWow) to enable nodes

authentication in a wireless sensor network. By integrating the memory cost, we have a better view of the overall area needed for GPS. Two of our cores are compatible with the platform constraints.

We have shown the benefits of having a fixed key for the implementation of GPS. In return, an adversary may exploit these features to mount *side-channel attacks* (SPA, DPA...). While not considered in this work, it will be interesting to analyze how resistant are our implementations to these attacks.

We focused on GPS but our results impact the implementation of more cryptosystems. In term of hardware architecture, GPS is very similar to WIPR [15] and BlueJay [16]. These two cryptosystems are based on an early paper of Shamir [17] which introduced *randomized multiplication cryptosystem*. The cores of these proposals are an adder and a multiplier as for GPS. Our solutions can be adapted to implement WIPR, BlueJay and Shamir's scheme. In a future work, we will compare the implementations of these different schemes.

REFERENCES

[1] T. Eisenbarth, S. S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of Lightweight-Cryptography Implementations," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 522–533, 2007.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] B. Wyseur, "White-Box Cryptography," in *Encyclopedia of Cryptography and Security (2nd Ed.)*, H. C. A. van Tilborg and S. Jajodia, Eds. Springer, 2011, pp. 1386–1387.

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

[13] 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.

[14] 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.

[15] Y. Oren and M. Feldhofer, "A low-resource public-key identification scheme for RFID tags and sensor nodes," in *ACM Conference on Wireless Network Security - WISEC 2009*. Zurich, Switzerland: ACM, March 2009, pp. 59–68.

[16] M.-J. O. Saarinen, "The BlueJay Ultra-Lightweight Hybrid Cryptosystem," in *Workshop on Special Aspects of Cyber Physical Systems - TRUSTED 2012*. San Francisco, CA, USA: IEEE, May 2012, p. To appear.

[17] A. Shamir, "Memory Efficient Variants of Public-Key Schemes for Smart Card Applications," in *Advances in Cryptology - EUROCRYPT '94*, ser. Lecture Notes in Computer Science 950. Perugia, Italy: Springer, 1994, pp. 445–449.