On area, time, and the right trade-off

Poschmann, A.; Robshaw, M. J. B.

2012

Poschmann, A., & Robshaw, M. J. B. (2012). On Area, Time, and the Right Trade-Off. Proceedings of the 17th Australasian Conference, ACISP 2012, 7372, pp.404-418.

<https://hdl.handle.net/10356/99703>

https://doi.org/10.1007/978-3-642-31448-3_30

© 2012 Springer-Verlag Berlin Heidelberg. This is the author created version of a work that has been peer reviewed and accepted for publication by Proceedings of the 17th Australasian Conference, ACISP 2012, Springer-Verlag Berlin Heidelberg. It incorporates referee's comments but changes resulting from the publishing process, such as copyediting, structural formatting, may not be reflected in this document. The published version is available at: [http://dx.doi.org/10.1007/978-3-642-31448-3_30].

Downloaded on 26 Apr 2023 20:27:53 SGT

On Area, Time, and the Right Trade-Off

A. Poschmann¹ and M.J.B Robshaw^{2*}

¹ Nanyang Technological University, Singapore
aposchmann@ntu.edu.sg

² Applied Cryptography Group, Orange Labs, France
matt.robshaw@orange-ftgroup.com

Abstract. Recently one of the most active fields of cryptography has been the design of lightweight algorithms. Often the explicit goal is to minimise the physical area for an implementation. While reducing area is an important consideration, beyond a certain threshold there is little point minimising area further. Indeed, it can be counter-productive and does not necessarily lead to the most appropriate solution. To provide a clear demonstration of this, we consider two lightweight algorithms that have been proposed for deployment on UHF RFID tags and which appear in a forthcoming ISO standard. Our results show that by choosing an implementation strategy that reduces but not necessarily minimises the area, very significant savings in time and substantial reductions to other physical demands on tag performance can be delivered. In particular, given the crucial importance of transaction time in the deployment of most contactless applications, our work illustrates that the most suitable practical implementation does not always conform to expectations.

1 Introduction

The challenging physical constraints posed by RFID tags have been a significant spur to cryptographic research. The world of RFID tag deployment is dominated by two operating frequencies; HF and UHF. Tags that communicate with an interrogator using HF (13.56 MHz) are well-established. Most familiarly we see these in public-transport applications and they will play a significant role in the proliferation of *Near Field Communication (NFC)* applications. Happily, the physical constraints for short-range HF-based tags are not onerous: sufficient power can be delivered to the tag to deploy standard cryptographic primitives like Triple-DES [23] and AES [24]. By contrast, the much cheaper and much smaller tags standardised by EPCglobal [4] communicate using UHF (860–960 MHz). As well as size and price, a significant advantage of UHF tags is that they can be read at a distance. For these tags, though, space and power consumption are at a premium.

The UHF RFID tag is a remarkable piece of engineering; not only are these passive devices small and cheap enough to be attached to millions of objects—for *track-and-trace* applications in the supply chain—but they must operate in

* The author gratefully acknowledges the support of the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

multi-tag and multi-reader environments with exceptional reliability. This success is spurring their wide-spread deployment and, at the same time, leading calls for an extension of their functionality. One particular application of interest is that of product authentication and calls to use cheap UHF RFID tags as part of an anti-counterfeiting solution are well-known [1, 17, 19]. But the favored long-term solution to the problem, that of providing dynamic cryptographic tag authentication, requires lightweight cryptographic algorithms and standardised solutions would be a plus. This consideration helps provide a focus to the work in this paper. Our goal is to consider the issue of area-time trade-offs and as targets we consider two parts of the forthcoming ISO/IEC 29192 standard [16] dedicated to lightweight cryptography.

1.1 Area and Time: The obvious trade-off?

Of course it is easy to say (and it is true) that there always exist trade-offs in implementation strategy. Indeed area and time offer the basis for the most well-known trade-offs in hardware implementation. Yet such a statement can sometimes mask considerable complexities. To those that rarely work at the hardware level, an area-time tradeoff often suggests that a factor t increase in area allows a t -fold parallelisation of algorithm components. This would yield a factor t reduction in processing time (also called *latency*) and a constant *area-time product* (AT-product). However when we consider the range of trade-offs available, the AT-product is rarely, if ever, constant since there are fixed-size components in any implementation. Further, for systems that result from an accumulation of optimisations, understanding the net gain for any given trade-off requires careful analysis.

The pioneering papers in the field of lightweight cryptography effectively defined the operating parameters for the field. With good justification, the primacy of area was singled out as the most influential parameter in an implementation. However many other factors effect the suitability of a solution in deployment and these factors are often set aside. For example, HB⁺ [18] was particularly lightweight in terms of the on-the-tag operations, however it was observed in [5] that the scheme would require more than 80 000 bits of communication between the tag and the reader which is plainly infeasible. Similarly, transaction time is rarely considered as a limiting factor in much of the academic work on RFID tags; yet for contactless applications with a multitude of tags passing a reader or portal at speed, this can be critical. Our goal in this paper, therefore, is to explore a range of implementation strategies for certain core technologies. By doing so we hope to highlight a more complete range of factors that determine the suitability of a solution for a particular application.

1.2 This paper

In this paper we consider the value of implementation strategies that might differ from the “minimum-area” approach. While our focus will be on area and time,

Table 1. Area-time trade-offs for the block cipher PRESENT with 80-bit keys. The area-time product is proportional to the energy consumption per bit encrypted.

		Datapath width				
		64	32	16	8	4
<i>scaleable</i>	S-layer	379.61	200.97	111.65	66.99	44.66
	key xor	170.88	85.44	42.72	21.36	10.68
	sub-total	550.49	286.41	154.37	88.35	55.34
<i>overhead</i>	MUXes	0.00	149.12	74.56	37.28	18.64
	counter	0.00	5.00	10.00	15.00	20.00
	sub-total	0.00	154.12	84.56	52.28	38.64
<i>fixed</i>	flip-flops	864.00	864.00	864.00	864.00	864.00
	counter	54.00	54.00	54.00	54.00	54.00
	sub-total	918.00	918.00	918.00	918.00	918.00
Area (GE)		1 468.49	1 358.53	1 156.93	1 058.63	1 011.98
Time (clk)		31	65	129	258	516
Area-Time product		45 523	88 304	149 244	273 127	522 182
Factor increase		1.0	1.9	3.3	6.0	11.5

we will keep our eye on other factors. Maximum and average power consumption are important issues, particularly for UHF RFID tags that will be read at a distance. For the lightweight designs we consider in this paper, power consumption is dominated by static power consumption which is proportional to the physical area of the implementation. This suggests that we would always be particularly interested in minimum-area implementations. However an important consideration, particularly say for battery-powered sensor nodes, can be the energy consumption of an implementation. And since energy is given by the product of power and time, energy consumption will be effectively proportional to the area-time product.

To illustrate the issues in this paper, we have decided to concentrate on two algorithms that feature in ISO 29192, a forthcoming standard dedicated to lightweight cryptographic techniques. This multi-part document covers block ciphers (part two), stream ciphers (part three), and asymmetric techniques (part four) and we decided to consider PRESENT and CRYPTOGPS here. It is possible that the reader will find the paper somewhat unbalanced, with more space being attributed to the trade-off in CRYPTOGPS than to PRESENT. However this is a direct result of the more complex trade-off offered by CRYPTOGPS which takes some analytic and implementation effort to fully understand.

To compare the area of different implementations, it is typical to use the concept of the *gate equivalent (GE)*. The physical area of an implementation is divided by the size of a **nand** gate to give, what is intended to be, a broadly technology-neutral estimate of the size of an implementation. Since there can be significant variations in the area reported for different technologies it is not perfect, but it nevertheless remains a reasonable measure to use.³

³ Provided we avoid making claims on the basis of a difference of a few hundred GE.

Fig. 1. Area (GE), time (cycles), and AT-product (scaled down by a multiplicative factor of 300 for convenience) for different implementations of 80-bit PRESENT.

2 Area and Time for PRESENT

PRESENT is a compact block cipher with a classical SPN structure [3]. Designed to take either 80- or 128-bit keys, its simple regular structure means that the cipher offers a wide range of implementation options. The net results of some of these trade-offs are presented in Table 1. As can be seen from the table, the components in an implementation can be separated into three classes; components that are scaleable, components that are fixed-size, and any overheads that are required to deal with a specific implementation strategy. All the figures given in Table 1 were derived using *Synopsys DesignCompiler A-2007.12-SP1* to synthesize the VHDL code to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3* [31].

The results in Table 1 are worth considering in some detail. When considered solely in terms of area, there is not much to choose between the different implementations; the area lies between 1 000 and 1 500 GE, a variation of 50%. Yet, at the same time, the time to encrypt 64 bits varies by a factor of more than 16 and energy consumption per bit can vary by a factor of more than 11. This is illustrated in Fig. 1. It can therefore be hard to decide which approach would be most suitable for a given application. But it seems almost certain that the smallest implementation (of around 1 000 GE) will only be suitable on an

Table 2. A headline comparison of the efficient implementation of the two block ciphers that feature in ISO 29192-2 and the AES.

PRESENT (this paper)			CLEFIA ([30, 2])		
Area (GE)	Time (clk)	AT- product	Area (GE)	Time (clk)	AT- product
1 468	31	45 523	6 050	18	108 900
1 359	65	88 304	5 490	36	197 640
1 157	129	149 244	2 678	176	471 328
1 059	258	273 127	2 594	192	498 048
1 012	516	522 182	2 488	328	816 064
			AES ([22, 12])		
			Area (GE)	Time (clk)	AT- product
			3 100	160	496 000
			2 400	210	504 000

exceptional basis. Instead it is likely that the implementation of around 1 500 GE will provide the best trade-off; in short the largest implementation.

For the sake of completeness we provide some area-time tradeoffs for the implementation of CLEFIA [30, 2] which also appears in ISO 29192-2 and AES [24] which does not. These are 128-bit block ciphers that use a 128-bit key and it is, therefore, only to be expected that they will both require more space for an implementation. Given these different operational parameters it is quite difficult to make a meaningful comparison between PRESENT and CLEFIA or AES. For these last two ciphers the implementation results, by other authors, are presented in Table 2. Since different synthesis tools and technologies have been used, a close comparison between CLEFIA and AES cannot be made. However it is interesting that the trade-offs available to CLEFIA and AES appear to be broadly comparable and these figures suggest that CLEFIA and AES are likely to offer very similar implementation characteristics in terms of area, time, and energy efficiency.

3 The Trade-off for CRYPTOGPS

CRYPTOGPS is a commitment-challenge-response scheme that features in part 4 of ISO/IEC 29192. It is due to Girault, Poupard, and Stern. and the security is well-established in the literature [6, 10, 29]. Several variants have previously been standardised [15] and, over the years, several optimisations [9, 11, 15] have been proposed. Since our focus here is on the implementation rather than the specifics of the scheme, we defer a detailed description to Appendix A.

Instead, our interest lies in a special trade-off that is available to CRYPTOGPS. To see this, it suffices to list the actions on the tag where x_i is a short commitment computed at the time of tag manufacture from r_i , where r_i is a sufficiently long pseudo-random string, c is a challenge from the reader, and s

is the tag secret key. To understand the relation between these quantities and their appropriate sizes, see the appendices.

CRYPTOGPS Tag Actions :

- (a) **send commitment** x_i
- (b) **receive challenge** c
- (c) **compute** $y_i = r_i + sc$
- (d) **send response** y_i

Step (c) is where the tag computation takes place and consists of two components. First we are required to compute $y_i = r_i + sc$. This simple computation involves integer addition and integer multiplication; there is no modular reduction and we will denote this version the **mult** variant. In addition to this calculation, optimisations outlined in ISO 9798-5 [15] suggest that r_i be re-generated on-the-fly at the time of use (instead of being stored). To do this we can use a pseudo-random number generator, perhaps PRESENT in a suitable mode of use.

While avoiding a modular reduction is a major step towards UHF tag deployment, another optimisation has sought to overcome the cost of the multiplication itself. This is done by means of what is termed a *Low Hamming Weight challenge* [9]. Here the interrogator chooses a challenge that is longer than usual but which has a very low Hamming weight. Since there are few ones in the challenge and they can be judiciously spaced; the multiplication ($s \times c$) on the tag is turned into a modest number of additions. This gives the potential to further optimise the on-tag computation and we will denote this version the **lhw** variant.

It is instructive to look at some sample parameter values, offering the same security, which highlight some of the essential differences between the **mult** and **lhw** variants. Even though we must increase the length of the challenge c in the **lhw** variant, to maintain the same security level for a comparable **mult** variant, the challenge is sparse and allows a variety of compact representations (see [21, 28] for more details). In fact it turns out that the cost of transmitting the challenge for the **mult**-variant or the compressed challenge for the **lhw**-variant is the same, though there is a very minor penalty in decoding the compressed challenge on the tag. We summarise the basic trade-offs below for some sample parameter sets. The contrast between the two variants is clear. By avoiding an integer multiplication (in the **lhw** variant) we need a longer c , which means we need a longer r_i which takes longer to generate. This increases the time required to compute the response y_i as well as the time required to transmit it.

	mult -variant	lhw -variant
$\{ s , k , c \}$	$\{160, 80, 36\}$	$\{160, 80, 1179\}$
communication of c (bits)	36	36
on-tag computation required	add, mult	add
length of r_i (bits)	276	1419
communication of y_i (bits)	276	1419

Note that this is a very different area-time trade-off than for Sect. 2. In the case of PRESENT the components of the computation remained unchanged;

<i>variant</i>	<i>w/o</i> PRNG		<i>with</i> PRNG			
	synthesized [20, 21]		synthesized [28]		fabricated [28]	
	<i>area</i> (GE)	<i>time</i> (cycles)	<i>area</i> (GE)	<i>time</i> (cycles)	<i>area</i> (GE)	<i>time</i> (cycles)
1-bit	317	1 088	-	-	-	-
4-bit	-	-	2 143	9 319	2 403	9 319
8-bit	431	136	2 433	724	2 876	724
16-bit	900	68	-	-	-	-

Fig. 2. The performance profiles offered by different architectures, denoted *variant*, for existing work on the **1hw**-variant of CRYPTOGPS. Work on the left [20, 21] is solely concerned with the computation of y_i , while work on the right [28] also incorporates the regeneration of r_i .

throughout we used the same S-boxes and the same diffusion layer but these were packaged in different ways. In the case of CRYPTOGPS however, the operation itself is changed from a multiplication to an addition and the trade-off is far more complicated.

In all previous implementation papers of CRYPTOGPS the single goal of minimising area meant that the **1hw**-variant was used. The range of implementations included FPGA [8] as well as synthesized [20, 21] and fabricated [28] ASICs. The most useful comparison point is that reported in [28] which offers a full *fabricated* implementation using 0.25 μm technology. There the **1hw**-variant was implemented with PRESENT in OFB mode as a way of regenerating the r_i . Using two alternative implementations of PRESENT, serial and round-based, the results showed that the full cost of CRYPTOGPS would likely be bracketed by 2876 GE and 724 cycles as the most time-efficient implementation and 2403 GE but 9319 clock cycles for the most space-efficient variant. The headline comparison between all this prior work is provided in Fig. 2. However, it is the **mult** variant that is likely to be more appealing in practise, as we will see in the next section.

3.1 Implementations of the mult-variant

Implementations of the **1hw**-variant showed that CRYPTOGPS was conceptually feasible on passive UHF RFID tags, particular if we focus on the key indicators of area and power consumption, see Fig. 2. However even then, the variant in Fig. 2 most likely to be preferred in practice is the one with the largest area requirement; yielding a response in 724 cycles with an area of 2876 GE. A closer look at the implementation costs [28] is revealing, and motivates our exploration of the area-time trade-off for CRYPTOGPS.

As we can see from Fig. 3 there are four main components to the implementation; PRESENT for the regeneration of r_i , the addition used in the CRYPTOGPS response computation (called **Addwc** in [28]) the controller, and some storage. We observe that the bulk of the implementation cost is due to PRESENT. This is not a problem with PRESENT itself, but rather an indication of the very low com-

Total Implementation: 2876 GE					
PRESENT		Addwc		Controller	
[GE]	%	[GE]	%	[GE]	%
1751	60.9	60	2.1	905	31.5
				159	5.5

Fig. 3. The area breakdown for the CRYPTOGPS implementation of [28].

putation overhead incurred when supporting the `1hw`-variant of CRYPTOGPS. While the computation time of 724 cycles is reasonable, amounting to seven ms if the digital component is clocked at 100 KHz, we note from Sect. 3 that the size of the response y_i is quite large. The implications would depend on the application and use-case, but low-end communication rates provided in data sheets [26] suggest that returning y_i for the `1hw`-variant could take anywhere up to five times longer than its computation. This potential drawback could be avoided by moving to a version of CRYPTOGPS that uses multiplication. There will be a cost since the space required to support `mult` will certainly be several multiples of that required to support integer addition. However, since 61% of the `1hw` implementation is dominated by PRESENT (see Fig. 3) then the impact on the *overall* cost of implementation in moving to `mult` will not be significant. Indeed, we will show that the additional total area overhead should be only around 20%.

Implementing multiplication. In this, and later sections, we will refer to some specific operand sizes and these correspond to the parameter set described in Appendix B. The computation at the heart of CRYPTOGPS is $y_i = r_i + sc$. In the `mult` variant, the computation of sc can be done in a variety of ways and the most important parameter is the word-size used for this multiplication. Do we perform a multiplication bit-by-bit or do we build the multiplication out of a series of n -by- n -bit operations? In addition to the impact on the area, our choice has some timing and latency implications. An indication of the trade-offs involved for different n are given in Table 3 where we give the area and the number of cycles to generate the least significant 16, 32, 48, and 64 bits of a product respectively. We can assume that one factor, of length 36 bits, is variable while the other has fixed length 160 bits.

Combining with PRESENT. To find the most suitable implementation we need to recall that when we compute sc we are also required to compute r_i . Using PRESENT in OFB mode, *i.e.* following [28], means that after each encryption operation (32 clock cycles), 64 bits of r_i are available. It makes sense for the relevant 64 bits of sc to be available at the same time, so that they can be immediately added to the lower parts of the product as it is generated rather than having to store any intermediate values.

It turns out for our choices of n that `mult` requires more clock cycles than PRESENT. As a consequence we tried to reduce the timing requirements of `mult` while keeping the area requirements low. We chose a basic SHIFT-AND-ADD

Table 3. Time (`clk`) to generate b bits when using n -by- n -bit operations within the multiplication. Area estimates (GE) are provided at the foot of the table.

bits generated (bits)	size of operand (bits)				
	$n = 4$	$n = 8$	$n = 16$	$n = 32$	$n = 36$
$b = 16$	112	44	16	16	16
$b = 32$	256	116	52	32	32
$b = 48$	400	188	88	52	48
$b = 64$	544	260	124	68	64
<i>area (GE)</i>	820	850	950	1 020	1 040

algorithm, where the least significant n bits of s are added to an intermediate result if the i^{th} bit of the challenge, bit c_i , is one; zero is added otherwise. Then the intermediate value is shifted by one bit to the right and the next bit of the challenge is used to determine whether the same chunk of s or zero is added.

Once all bits of c are processed, the next n bits of s are used and the procedure repeats. In this way the first n output bits sc_i are available after i clock cycles. After that it takes 36 clock cycles for each consecutive n output bits until all 160 bits of s have been processed. At this time all 196 bits of sc are available.

The combination of a SHIFT-AND-ADD multiplier and round-based PRESENT has the advantage that parts of sc can be immediately added to r_i as they become available, while the time to compute the next part of sc (36 clocks) is also used to generate the next 64 bits of r_i (which requires only 32 clocks) in parallel.

As a side-issue, it is interesting to note that this discussion really helps to highlight the role and impact of latency in a design. While the time required to generate r_i should drop by a factor close to five—from 23 iterations of PRESENT to five iterations—the overall time to compute the response y_i will only drop by a factor of two. This is entirely due to the larger timing requirements carried by `mult` when compared to PRESENT.

3.2 Implementation results for $|c| = 36$

For our implementation of CRYPTOGPS we followed the work of [28] as closely as possible. We are not in a position to go through the fabrication process, but we can get good estimates on the performance of our implementations from synthesis. We used *Synopsys DesignCompiler A-2007.12-SP1* to synthesize our VHDL code to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3* [31], which is based on the *UMC L180 0.18 μ m 1P6M* logic process with a typical voltage of 1.8 V. We used *Synopsys Power Compiler* version *A-2007.12-SP1* to estimate the power consumption of our ASIC implementations. For synthesis and for power estimation we advised the compiler to keep the hierarchy and use a clock frequency of 100 KHz.

The input and output port size of CRYPTOGPS, MULT, PRESENT and the full-adder component is denoted by IO (see Fig. 4(a)). It is typically reasonable

Table 4. The area of the different components under two implementation strategies for PRESENT and multiplication.

	PRESENT		MUL		Controller		Adder		Total
	[GE]	%	[GE]	%	[GE]	%	[GE]	%	[GE]
$n = 36, IO = 32$	1 689	54.8	1 034	33.5	185	6.0	175	5.7	3 083
$n = 4, IO = 4$	1 651	58.3	832	29.4	319	11.3	28	1.0	2 830

to choose $IO = n$, but for our first implementation we used an operand size of $n = 36$ (since this should give the fastest implementation). However 36 is not a common bus width and we used $IO = 32$.

The challenge c is variable and so we save c in a shift-register that can operate in two ways: 1) as a bit-serial shift-register that *rotates* the content by one bit to the right; 2) as an IO bit shift-register that *shifts* the content by IO bits to the right. When $|c|$ is not a multiple of IO the least significant bits are discarded.

Since s can be fixed we chose to hardwire it, which gives a saving of 160 flip-flops. The appropriate part of s is chosen by a multiplexer, where we pad the last part of s with zeros when n does not divide $|s|$. The part in question is AND-ed with an n -bit replication of the least significant bit of the challenge register and this serves as one input to an n -bit full-adder. The sum (including the carry overhead) is stored in an $(n + 1)$ -bit register and the n most significant bits serve as the second input to the adder. The least significant bit is stored in a bit-serial shift-register of length $IO + |c| - n - 1$ bits and the IO least significant bits serve as the output of this component which is ready every 32 clock cycles.

Recall that the 36 most significant bits of sc are ready at the same time, when all 160 bits of s have been processed. So to have a balanced design we chose to keep the output frequency and to save area at the cost of 29 additional clock cycles. In our implementation (depicted in Fig. 4(b)) y_i is available after 257 cycles. Power estimates for a frequency of 100 KHz, at a supply voltage of 1.8 V and using the smallest wire-load model (for circuits of about 10K GE) are $3.45\mu\text{W}$. The total area requirement is 3 083 GE with a breakdown provided in Table 4. The overhead for PRESENT, when compared to the implementation in [3], is due to additional multiplexers that are required to a) interface between PRESENT and ADDER, and to b) feedback the ciphertext (OFB mode).

Our second implementation aimed at minimal area while keeping an eye on the time, thus we choose an operand size of $n = IO = 4$ for the multiplication. Compared to our first implementation we used a different strategy to reduce the processing time. As soon as the multiplication has finished, the least significant bit shift register is stopped by a gated clock. The required part of sc is selected by a multiplexer, and this allows us to output one chunk in every clock cycle without introducing additional latency. This allows us to save 315 clock cycles at a cost of 63 GE for an additional multiplexer and a more complex control logic.⁴

⁴ It is not obvious to what extent the 134 GE increment of the controller module is caused by this design decision and not by the additional counters required *etc.*

Table 5. Comparison of time, area, and area-time (AT) product of **mult** and **lhw** variants [28, 27]. Extrapolated figures are marked with *.

Variant	$ c $	(n, IO)	Time (clk)	Area (GE)	AT product
mult	36	(4,4)	1 533	2 830	4 338 811
		(8,8)	799*	2 857*	2 282 991*
		(16,16)	432*	2 980*	1 287 338*
		(36,32)	257	3 083	792 311
	64	(4,4)	2 660*	3 138*	8 346 936*
		(8,8)	1 362*	3 177*	4 327 496*
		(16,16)	713*	3 300*	2 352 864*
		(32,32)	389*	3 424*	1 331 917*
lhw	1 179*	(8,8)	880*	2 556*	2 249 280*

In this implementation (depicted in Fig. 4(c)) y_i is available after 1 533 cycles. The total area is 2 830 GE (Table 4) with $3\mu\text{W}$ an estimated power consumption.

3.3 Estimations for $|c| = 64$

There is growing interest in a CRYPTOGPS variant with a 64-bit challenge. In this case r_i has a length of 304 bits, which still only requires 5 iterations of PRESENT, but outputting additional 28 bits (compared to the $|c| = 36$ variant) imposes a single figure clock cycle overhead. The true expenses lie in the multiplication part, as the time required for computing the product grows linearly with the challenge size $|c|$. The area increment is less severe, as the challenge size only impacts two register lengths. Hence the overhead comprises of 56-bits of additional storage (299 to 336 GE). We have included timing and area estimates for this variant in Table 5.

4 Area and Time for CRYPTOGPS

While it will depend to some extent on the application, it seems that the **mult**-variant of CRYPTOGPS has been neglected in the cryptographic literature. While it is not straightforward, we can extrapolate the results in [28] to make a comparison. In that work the **lhw** challenge consisted of a 847-bit string that had a Hamming weight of five. These were the parameter choices specified in the original proposal by Girault and Lefranc [9]. Taking account of other considerations [7], we are interested to explore a different parameter set. However the net result of the changes to the figures given in [28] is surprisingly minor. A simple way of achieving a challenge space equivalent to 36 bits in the **lhw**-variant is to move to a slightly longer challenge of Hamming weight six. This leads to the sample parameter choices that are given in Appendix B. The power consumption will be effectively unchanged, as will the area for the implementation, though the computation time for y_i will increase from 724 cycles to an estimated 880

cycles⁵ due to the longer generation time for r_i . The same implementation of CRYPTOGPS in [28] has been synthesized to the same standard-cell library used⁶ in [27] allowing us to fairly compare area figures.

Using `mult` instead of `lhw` provides an interesting—but sophisticated—trade-off for CRYPTOGPS. Table 5 shows a comparison of our implementations using `mult` with the `lhw` variant reported in [28, 27] with regards to time (in clock cycles), area (in GE), and the area-time (AT) product. All metrics are considered to be better the smaller they are. It is easy to see that the choice of $n = 4$ is not very useful, as it is 11% larger and 75% slower than `lhw`, resulting in an AT product that is nearly twice as bad. However, that changes with a growing operand size. An operand size of $n = 8$ will already yield a similar AT product as the `lhw` variant, and $n = 16$ will result in an AT product that is 43% smaller than the `lhw` variant. The real advantage of `mult` lies in larger operand sizes, such as $n = 32$, which yields an AT product that is only a third of the `lhw` variant, indicating energy savings in the same range and a much faster processing time.

5 Conclusion

In this paper we have highlighted the importance, and the very different results, that arise from different implementation strategies for lightweight cryptography. While area and time offer a fundamental trade-off, the results of pursuing such a trade-off are not always so obvious.

As a first example we considered PRESENT where the different trade-off points are reasonably straightforward to establish. And while many papers on security solutions for RFID would suggest that we take the minimum-area implementation of PRESENT, in reality it is hard to see an application where this would really be the preferred solution. Instead, a much more practical trade-off is obtained by taking the largest-area implementation of the ones we examined which takes one sixteenth the computation time and one eleventh the energy per bit of encryption. The moderate increase in area that is required will almost certainly be viewed as a reasonable cost.

This is further exemplified when we consider the typical strategy of minimising area in implementations of CRYPTOGPS. This is achieved by replacing the conventional multiplication in CRYPTOGPS with a long sparse multiplication, the so-called `lhw`-variant. However, we have shown in this paper that by using multiplication, and thereby accepting an increase in area, we appear to get a more reasonable trade-off. If we look purely at the relative costs of addition in `lhw` and multiplication in `mult`, then the area increases from 60 GE to over 1 000 GE, a factor of more than 16. The initial reaction would be to choose the `lhw`-variant. However the true additional cost of supporting multiplication—when considered in relation to the *entire* cryptographic component—is only around 20%. So by accepting such a modest increase to a low-area implementation, the

⁵ To generate 1 419 bits requires 23 iterations of PRESENT instead of the 19 used in [28].

⁶ Table 7.1 in [27] reports a higher cycle count since they also included the overhead from the *IO* handshake protocol; this has been excluded in [28] and here.

on-tag computation time can be reduced by more than 70%, the communication overhead can be reduced by 80%, and the total transaction time can be potentially reduced by more than 75%. These are significant performance savings, all at the same security level.

While we have deliberately focused on two algorithms that can be found in the forthcoming ISO standard 29192, our work carries broader lessons for the use of cryptography in constrained devices. In particular, while area is a vital metric, the minimum-area implementation is not necessarily the most useful in practice.

References

1. M. Aigner, T. Burbridge, A. Ilic, D. Lyon, A. Soppera, and M. Lehtonen. RFID Tag Security, BRIDGE white paper. Available via www.bridge-project.eu.
2. T. Akishita and H. Hiwatari. Very Compact Hardware Implementations of the Blockcipher CLEFIA. In *Proceedings of SAC 2010*, pages 2925–2928, IEEE, 2008.
3. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Proceedings of CHES '07*, volume 4727 of LNCS, pages 450–466. Springer, 2007.
4. EPCglobal. EPC Radio-Frequency Identity Protocols, Class-1 Generation-2 UHF RFID, Protocol for Communications at 860-960 MHz, version 1.2.0. October 23, 2008. Available via www.epcglobalinc.org.
5. H. Gilbert, M. Robshaw, and Y. Seurin. HB[#], Increasing the Security and Efficiency of HB. In N. Smart, editor, *Proceedings of Eurocrypt '08*, volume 4965 of LNCS, pages 361–378, Springer, 2008.
6. M. Girault. Self-certified public keys. In D.W. Davies, editor, *Proceedings of Eurocrypt '91*, volume 547 of LNCS, pages 490–497, Springer-Verlag, 1991.
7. M. Girault. Low-Size Coupons for Low-Cost IC Cards. In J. Domingo-Ferrer, D. Chan, and A. Watson, editors, *Proceedings of Smart Card Research and Advanced Applications*, pages 39–50, Kluwer Academic Press, 2001.
8. M. Girault, L. Juniot, and M. Robshaw. The Feasibility of On-the-Tag Public Key Cryptography. *RFIDsec 2007*, workshop record. Available via rfidsec07.etsit.uma.es/slides/papers/paper-32.pdf.
9. M. Girault and D. Lefranc. Public Key Authentication with One (Online) Single Addition. In M. Joye and J.-J. Quisquater, editors, *Proceedings of CHES '04*, volume 3156 of LNCS, pages 967–984, Springer-Verlag, 2004.
10. 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, pages 463–487, Springer, 2006.
11. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values Used in Identification Schemes. In Y. Desmedt, editor, *Proceedings of Crypto '94*, volume 893 of LNCS, pages 202–215, Springer, 1994.
12. P. Härmäläinen, T. Alho, M. Hännikäinen, and T. D. Härmäläinen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In *DSD*, pages 577–583, 2006.
13. G. Hofferek and J. Wolkerstorfer. Coupon recalculation for the GPS Authentication Scheme. In G. Grimaud and F.-X. Standaert, editors, *Proceedings of Cardis 2008*, volume 5189 of LNCS, pages 162–175, Springer, 2008.

14. M. Hutter and C. Nagl. Coupon recalculation for the Schnorr and GPS Identification Scheme: A performance evaluation. In proceedings of RFIDSec 2009. Available via www.cosic.esat.kuleuven.be/rfidsec09/.
15. ISO/IEC 9798: Information Technology – Security Techniques – Entity Authentication – Part 5: Mechanisms using Zero-Knowledge Techniques. Available via www.iso.org.
16. ISO/IEC 29192-4: Information Technology – Security Techniques – Lightweight Cryptography – Part 4: Public key techniques. Committee Draft.
17. J. Jenkins, P. Mills, R. Maidment, and M. Profit. Pharma Traceability Business Case Report. BRIDGE white paper, May 2007. Available via www.bridge-project.eu.
18. A. Juels and S.A. Weis. Authenticating Pervasive Devices With Human Protocols. In V. Shoup, editor, *Advances in Cryptology - Crypto 05*, Lecture Notes in Computer Science, volume 3126, pages 293–198, Springer-Verlag, 2005.
19. M. Lehtonen, J. Al-Kassab, F. Michahelles, and O. Kasten. Anti-counterfeiting Business Case Report. BRIDGE white paper, December 2007. Available via www.bridge-project.eu.
20. M. McLoone and M.J.B. Robshaw. Public Key Cryptography and RFID. In M. Abe, editor, *Proceedings of CT-RSA '07*, volume 4377 of LNCS, pages 372–384, Springer, 2007.
21. M. McLoone and M.J.B. Robshaw. New Architectures for Low-Cost Public Key Cryptography on RFID Tags. In *Proceedings of SecureComm '05*, pages 1827–1830. IEEE Computer Society Press, 2007.
22. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In K. Paterson, editor, *Proceedings of Eurocrypt 2011*, volume 6632 of LNCS, pages 69–88, Springer, 2011.
23. National Institute of Standards and Technology. SP-800-67: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Revision 1, January 2012. Available via csrc.nist.gov.
24. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001. Available via csrc.nist.gov.
25. National Institute of Standards and Technology. FIPS 180-4: Secure Hash Standard, February 2011. Available via csrc.nist.gov.
26. NXP Semiconductors. UCODE EPC G2 Data Sheet. 2006. Available via www.nxp.com.
27. A. Poschmann. *Lightweight Cryptography - Cryptographic Engineering for a Pervasive World*. Number 8 in IT Security. Europäischer Universitätsverlag, 2009. Published: Ph.D. Thesis, Ruhr University Bochum.
28. A. Poschmann, M.J.B. Robshaw, F. Vater, and C. Paar. Lightweight Cryptography and RFID: Tackling the Hidden Overheads. In D. Lee and S. Hong, editors, *Proceedings of ICISC 2009*, volume 5984 of LNCS, pages 129–145, Springer, 2010.
29. G. Poupard and J. Stern. Security Analysis of a Practical “on the fly” Authentication and Signature Generation. In K. Nyberg, editor, *Proceedings of Eurocrypt '98*, volume 1403 of LNCS, pages 422–436. Springer-Verlag, 1998.
30. T. Sugawara, N. Homma, T. Aoki, and A. Satoh. High-performance ASIC implementations of the 128-bit block cipher CLEFIA. In *Proceedings of ISCAS 2008*, pages 2925–2928, IEEE, 2008.
31. Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18 μm , July 2004.

Appendix A: The CRYPTOGPS scheme

The typical description of CRYPTOGPS in Fig. 5 incorporates several optimisations. Among them is a storage/computation trade-off, the use of what are called *coupons*, and this has been discussed in a variety of papers [7, 11]. A form of pre-computation that is stored on the tag, the small additional cost in memory (see Appendices B and C) is more than offset by the removal of all elliptic-curve operations from the tag, though the security of the scheme remains dependent on the elliptic-curve problem. At the time of manufacture t coupons, say (r_i, x_i) for $1 \leq i \leq t$, are computed. Under the assumption that we are required to use more than a handful of coupons, ISO 9798-5 proposes to avoid storing the full coupon (r_i, x_i) but instead to store a partial coupon x_i . We would use a compact *pseudo-random number generator* PRNG to generate r_i at the time of tag manufacture (or personalisation) and to re-generate r_i on the tag when needed.

Appendix B: Preferred parameter sets for CRYPTOGPS

Following the example set by many initiatives in the field of RFID tags, we aim for a basic security level of 80 bits. This means that the CRYPTOGPS secret s will be 160 bits long and when we come to generate r_i at the time of tag initialisation, and again during authentication, we will use PRESENT in OFB mode with an 80-bit key k . The keys s and k are fixed for the life of the tag while the IV used to begin OFB encryption will be variable (as required by any realistic CRYPTOGPS implementation). The form of the IV would likely contain some tag and/or coupon identifying information, but this is essentially an issue for the application architecture.

Like other work in the field, our starting point is a 32-bit random challenge from the interrogator. At the same time we use the work of Girault and Stern [7, 11] to reduce the size of the coupons that we need to store. For this, coupon generation uses a hash function, using say a member of the SHA-2 family [25], to reduce the elliptic curve point $r_i P$ to a shorter string which can, in turn, be further truncated [7].

The product sc , when using the mult-variant of CRYPTOGPS, will be 196 bits in length. So following the requirements of CRYPTOGPS [10] each r_i needs to be $196 + 80 = 276$ bits long. To regenerate r_i requires five iterations of PRESENT.

Appendix C: On the use of coupons

The use of coupons is not to everyone's taste and, indeed, they are not suitable to all use-cases. However limited-use tokens are familiar in a wide range of applications from pre-paid telephone cards to public transport ticketing. It might even be argued that coupons are ideally suited to many RFID applications where tags might be read 10 to 20 times and then either discarded or recommissioned. Certainly with a coupon size of 48 bits, see Appendix B, storing 10 or even 20 coupons is unlikely to be a serious cost in the coming years.

Some commentators are concerned that coupons could be consumed in a denial-of-service attack, *i.e.* by an attacker that maliciously exhausts coupons on a target RFID tag. While this is clearly true, the benefit to an attacker of such a time-consuming attack that needs to be repeated on tag-by-tag basis is rarely articulated. Just because it *can* be done, it doesn't mean that it *will* and it is hard to see any business case or material advantage to the adversary.

In response to the limited number of coupons, there has been some work regarding on-the-tag coupon regeneration [13, 14] though this does not seem to be realistic in deployment. By contrast more practical approaches have considered alternative ways of reloading coupons and one important observation is that coupons do not need to be carried on the tag at all; they can be delivered directly to the reader in the same way that we might deliver a public key. But all this takes us too far from the focus of the paper.

(a) Top-level hardware architecture of CRYPTOGPS.

(b) mult implementation with $N = 36$ and $IO = 32$.

(c) mult implementation with $N = 4$ and $IO = 4$.

Fig. 4. Hardware architectures of CRYPTOGPS and mult.

Tag	Reader
ELLIPTIC CURVE SYSTEM PARAMETERS	
Curve $\mathcal{C}$, point P	Curve $\mathcal{C}$, point P
KEYS	
Secret key $s \in_R \{0, 1\}^\sigma$	Pub. key $V = -sP$
Secret key $k \in_R \{0, 1\}^\kappa$	
COUPON PRE-COMPUTATION WITH PRG	
For $1 \leq i \leq t$ Let $r_i = \text{PRNG}_k(i)$ where $ r_i = \rho$ Set $x_i = \mathbb{H}(r_i P)$ Store coupon x_i	
PROTOCOL USING ON-TAG PRNG	
At time i fetch $x_i \xrightarrow{x_i}$ $\xleftarrow{c}$ Pick $c \in_R \{0, 1\}^\delta$ Gen. $r_i = \text{PRNG}_k(i)$ $y_i = r_i + (s \times c) \xrightarrow{y_i} \mathbb{H}(y_i P + cV) \stackrel{?}{=} x_i$	

Fig. 5. Overview of elliptic curve-based cryptoGPS using partially re-generated coupons. The PRNG can be implemented using, say, PRESENT while H denotes a hash function that is only needed at the time of coupon generation and tag verification; it is not implemented on the tag itself. The parameters ρ , δ , σ , and κ denote bit lengths that can be adjusted to offer a range of security/performance trade-offs.