Coupon Recalculation for the GPS Authentication Scheme*

Georg Hofferek and Johannes Wolkerstorfer

Graz University of Technology,
Institute for Applied Information Processing
and Communications (IAIK),
Inffeldgasse 16a, 8010 Graz, Austria.

Georg.Hofferek@iaik.tugraz.at
Johannes.Wolkerstorfer@iaik.tugraz.at

Abstract. Equipping branded goods with RFID tags is an effective measure to fight the growing black market of counterfeit products. Asymmetric cryptography is the technology of choice to achieve strong authentication but suffers from its ample demand of area and power resources. The GPS authentication scheme showed that a coupon-based approach can cope with the limited resources of passive RFID tags. This article extends the idea of coupons by recalculating coupons during the idle time of tags when they are powered but do not actively communicate. This approach relaxes latency requirements and allows to implement GPS hardware using only 800 gate equivalents plus storage for 560 bytes. In the average case it has the same performance as the classical coupon-based approach but does not suffer its susceptibility to denial-of-service attacks.

1 Introduction

Radio frequency identification (RFID) is an emerging technology for optimizing logistic processes. Goods or pallets can be equipped with small and cheap RFID labels to give them an electronic identity. RFID labels can be read over air interfaces without direct line of sight. The main components of an RFID label are an antenna and an RFID tag. The minimalistic tag is a small chip containing an analog front-end which is connected to the antenna and a digital part. The chip receives its power for operation over the same antenna which is used for communication. Thus, the available power budget is very small. The functionality of tags is very often limited to basic operations like sending a unique ID upon request. More sophisticated tags may contain sensors or memories for storing more information about the product they are attached to.

A promising field of application for RFID tags is the authentication of goods to prove their genuineness. Providing unique IDs is a first step but does not

* The results presented in this article origin from the European Union funded FP7 project *Bridge* (IST-2005-033546).

solve the problem because the wireless air interface, which operates either on 13.56 MHz (HF) or around 900 MHz (UHF), can be easily eavesdropped. Thus, an attacker could obtain the UID of an original product and produce an infinite number of cloned tags having the same identity. Cryptographic authentication will inhibit such counterfeiting of goods. This is a fast growing market because the economical damage of counterfeit products exceeds $600 billion annually [1].

During the last years many schemes have been proposed to bring strong cryptographic authentication to RFID tags. These activities focused on efficient hardware implementation because RFID tags have fierce constraints regarding silicon area and power consumption. The first choice was implementing symmetric cryptography, which has small footprint. A major work by Feldhofer et al. demonstrated the feasibility of implementing AES on passively powered RFID tags [4]. They achieved an AES encryption with a circuit complexity of 3500 gate equivalents taking 1000 clock cycles. On 0.35 µm CMOS technology, the encryption draws an average power of only 5 µW, which is well below the requirements of HF tags.

Although symmetric cryptography can be implemented efficiently, it suffers from the key distribution problem. In order to authenticate a product, the verifier must know the secret key. Thus, symmetric cryptography is only applicable in closed systems but not in world-wide logistic processes where not all involved parties are known in advance and in particular not all of them are trusted. Asymmetric cryptography overcomes many shortcomings of symmetric cryptography but requires a lot more resources. Area requirements are at least three times higher and the computation time in terms of clock cycles is usually more than hundred times higher. Most research on asymmetric cryptography for RFID centered on elliptic-curve cryptography, which offers reasonable security for RFID systems with key lengths of 113 to 256 bits [16, 19].

The GPS authentication scheme, named after its authors Girault, Poupard, and Stern, offers the possibility to fill the gap between symmetric and asymmetric cryptography [5, 6]. GPS is a public-key-based zero-knowledge protocol which is similar to Schnorr's scheme but uses a composite RSA-like modulus n. GPS is standardized in ISO/IEC 9798-5 [10]. There are also variants of GPS that are based on elliptic-curve cryptography. One of the remarkable properties of the GPS scheme is the possibility to use coupons. Coupons are lists of values which are needed during authentication. These values do not depend on any input from the interrogator, so they can be precomputed. That shifts the major computational load from the time of protocol execution to e.g. production time of the tag. §2 will go into the details of the GPS protocol. At this point it is only of interest that resource-constrained RFID tags can compute GPS authentications by having the ability to store a few coupons (each roughly between 200–500 bits for reasonable parameters) and to compute (non-modular) integer operations (addition, multiplication) with operand sizes of 160–260 bits.

The coupon approach can be efficiently implemented in hardware [12, 13]. The promising results of [12, 13] should not conceal profound drawbacks of the approach. First, only a limited number of authentications is possible because

storage is costly. Precomputed coupons have to be stored in non-volatile memory at the time of personalization. RFID tags usually use EEPROM memory as non-volatile memory. Memory sizes are kept at minimum to keep the cost of the tags low. Thus, only a few to a dozen of coupons are reasonable. This opens the door for denial-of-service attacks. Any RFID interrogator can request a GPS authentication from a tag, and few requests are sufficient to exhaust all coupons. This type of attack is difficult to prevent because the tag has no notion of time to limit the number of requests per time interval, nor does it have the possibility to authenticate interrogators. In order to authenticate an interrogator the tag would need the same computational resources as the reader. This would nullify the advantages of the coupon approach.

The coupon-recalculation approach presented in this article addresses these drawbacks. When RFID tags have the possibility to compute fresh GPS coupons during their idle time, an unlimited number of authentications is possible. Moreover the storage requirements can be lowered to store just a few coupons. When the comprehensive number-theoretic computations are done in the idle time, and thus well ahead of the actual use of the result, the requirements for the hardware change completely: Latency of the computation is no longer a key issue. Furthermore the processing can take comparatively long because RFID tags are usually long in the (electro-)magnetic field of the reader in comparison to the actual interrogation time. This allows to rethink hardware architectures for multi-precision modular arithmetic radically. We will show that tiny multiply-accumulate structures are sufficient to compute strong public-key cryptography on RFID tags.

The remainder of the article details the GPS authentication scheme and the coupon-recalculation approach in §2. §3 sketches the proposed hardware architectures. §4 presents algorithmic details and architectural details of a digit-serial approach optimized for small footprint and low power consumption. §5 presents achieved results and §6 concludes the article.

2 GPS Authentication

2.1 Basic Algorithm and Parameters

GPS authentication is a zero-knowledge authentication protocol that allows small hardware implementations of the prover wanting to assure its identity. A thorough analysis of the GPS authentication scheme, including comparisons with similar schemes is given in [9]. There, mathematical properties and background information are given, along with security considerations and references to almost all other literature about GPS. This section will only present the most important facts. The GPS protocol is depicted by Fig. 1. The prover first computes a commitment x, which is sent to the verifier. After having received the verifier's challenge c, the prover calculates the response y, which can then be checked by the verifier.

The parameters ρ, δ, and σ determine the bit lengths of the random value r, the challenge c, and the secret key s, respectively. Common values are $\sigma = 160$,

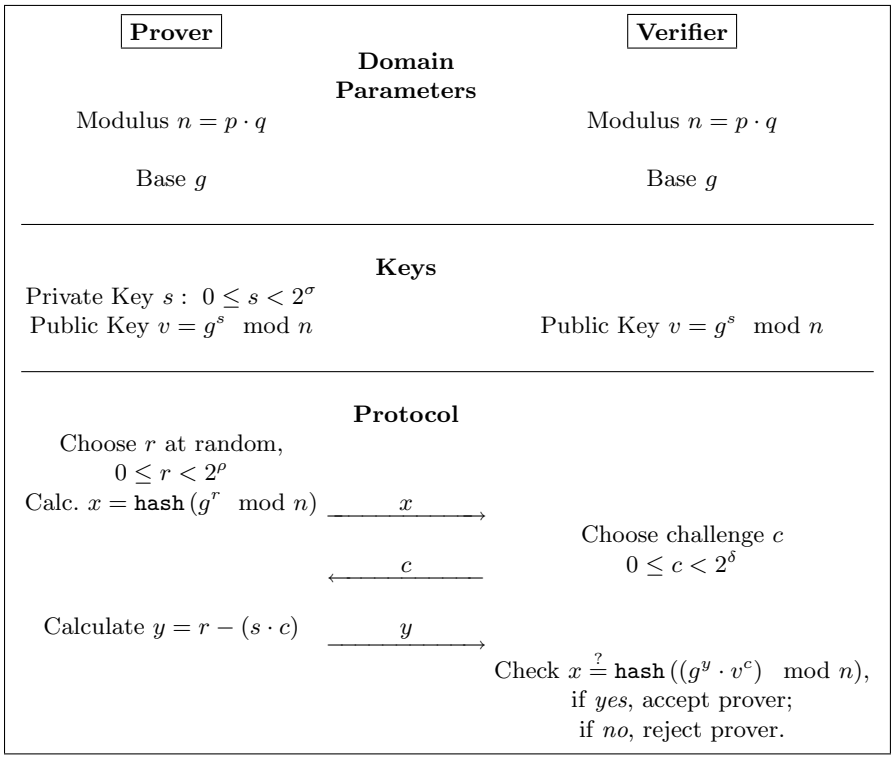

Fig. 1. Sketch of the basic GPS protocol, as standardized in [10].

$\delta = 20$, and $\rho = \sigma + \delta + 80 = 260$. The composite modulus n should be at least 1024 bits long. The integer subtraction in the last step can be substituted by addition if the public key is changed to $v = g^{-s} \mod n$ (cf. [15]). This simplifies the computation and allows an implementation of the prover without requiring signed numbers. The hash function shown in Fig. 1 is optional [10]. It mainly reduces bandwidth requirements for transmitting the commitment. In ultra-low footprint implementations it can be omitted.

2.2 Coupon Approach

The computation of the commitment x does not depend on any input from the verifier. This computation can already be done ahead of the actual protocol execution. This observation was made by Girault in [7]. This paves the way for the so-called *coupon approach*. A set of *coupons* (r_i, x_i) can be precomputed (e.g. at production time of the tag) and stored in non-volatile memory. When the protocol interaction starts, the prover selects a coupon (r_i, x_i) from memory and sends x_i to the verifier. After receiving the challenge c, the prover can compute the response $y = r_i + s \cdot c$. McLoone et al. use this approach in their implementations [12, 13]. They managed to implement the final integer operations on a

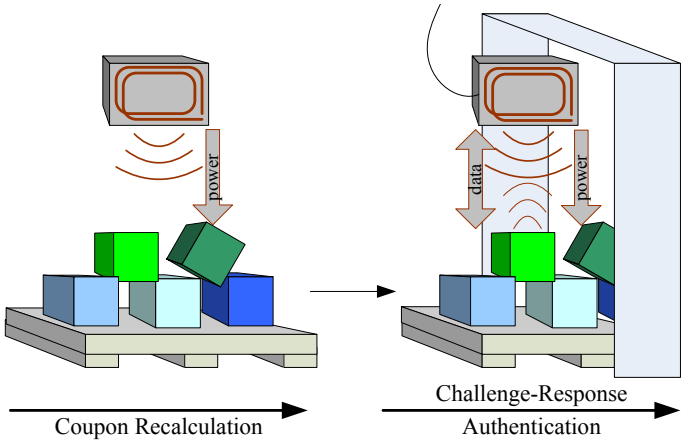

Fig. 2. Application scenario for the coupon recalculation approach.

circuit size smaller than 1000 gate equivalents in less than 150 clock cycles by exploiting a special variant of GPS based on challenges with a low Hamming weight (cf. [8]).

However, these implementations are susceptible to denial-of-service attacks. Once all coupons are used, the tag cannot authenticate itself any more. Due to the authentication being only unilateral a tag cannot determine whether an authentication request from a reader is warrantable or not. If a tag contained k coupons, an attacker could disable the tag by requesting k authentications. Giving tags the ability to (re-)calculate coupons prevents such denial-of-service attacks.

2.3 GPS Coupon Recalculation Approach for RFID

The coupon recalculation approach is an extension of the coupon approach. Whenever a tag is idle, but still supplied with power, it can use the time to compute new coupons to refill its coupon storage. RFID tags are powered rather long in comparison to the actual data transmission times. Tags are in the electromagnetic field of readers for seconds, while the protocol interaction completes within milliseconds. This also applies for high volume logistic processes like the one shown in Fig. 2.

In situations where it is very likely that all coupons have been used and thus need to be recalculated, RFID tags can be supplied with power before interrogation (see Fig. 2). Power supply requires only the transmission of the carrier frequency (e.g. 13.56 MHZ in HF systems). No sophisticated reader circuitry is necessary to drive the powering antennas. A simple oscillator is sufficient as interrogator circuit because neither modulation nor demodulation is needed.

The application scenario of coupon recalculation changes the requirements for the hardware radically. In contrast to previous asymmetric cryptographic hard-

ware neither latency nor throughput are of particular interest. Instead, hardware implementations can focus on minimizing silicon area and power consumption. The footprint of the circuit is of particular interest because the circuit size has also an linear impact on the power consumption. Thus, the smallest possible hardware suits the RFID requirements best. In order to keep the computation time within limits, the metric $A{\cdot}t{\cdot}P$, which weighs area, time, and power equally, was used to find an optimum hardware architecture.

Besides minimizing the area of the circuit it is also interesting to consider the maximum clock frequency f_{max}. Although RFID tags clock their digital circuitry at low clock frequencies to keep the power consumption low, high f_{max} can be used to accelerate coupon recalculation when a higher power budget is available. This is the case when a single good has to be authenticated. In such a situation the reader antenna is usually held next to the RFID tag. The power density of electro-magnetic fields are at least four times higher when halving the distance between the tag and the antenna. The digital circuit of RFID tags is almost ever optimized for worst-case situations at the far end of the reader field. Detecting higher power densities and exploiting them by increasing the clock frequency gives an RFID tag with GPS hardware the possibility to improve the average performance. Anyhow, next we will analyze hardware approaches assuming constant clock frequency.

3 Hardware Architecture

A hardware implementation of the recalculation approach can focus on the power efficiency and on low footprint optimization. For computing a fresh coupon (r, x), a modular exponentiation $x = g^r \mod n$ has to be computed. The square-and-multiply algorithm is the algorithm of choice for hardware implementations, which computes $x = \prod_{i=0}^{\log_2 r} g^{ir_i}$.

3.1 Full-Precision Architecture

An approach often used for (high-speed) hardware is to use a bit-serial full-precision multiplication to compute the modular multiplications and squarings. Many implementations even use digit-serial approaches to improve latency [3]. Most hardware implementations use Montgomery multiplication [14] instead of normal integer multiplication and subsequent modular reduction. Montgomery multiplication simplifies modular reduction. Bit-serial multiplication schedules one operand at full precision, while the other one is processed bit by bit. The disadvantage of this approach is that at least five full-precision registers are necessary. Three registers are needed for the multiplication: two for the input values, one for the (intermediate) result; one register is needed to store an auxiliary variable which is necessary for implementing the square-and-multiply algorithm, and one register is needed to store the modulus. Assuming a modulus of 1024 bits, this would mean that a full-precision architecture would need at least 5120

flip-flops, or more than 30 000 gate equivalents. After adding the necessary components for partial-product generation and accumulation, the complete datapath of a full-precision arithmetic unit requires approximately 50 000 gate equivalents (not counting non-volatile memory for storing domain parameters). When sticking to full-precision architectures, there are no more significant improvements to be made concerning the circuit's size. The size is mainly determined by the storage requirements, which depend on the size of the modulus. It is thus reasonable to concentrate on digit-level arithmetic architectures that operate on smaller word sizes, which will be discussed in the next section.

3.2 Digit-Level Architecture

Bit-serial or digit-serial hardware architectures strive for improving performance. In the recalculation approach, performance is of secondary interest. This allows to use hardware architectures with smaller footprint and lower power consumption. Multiply-accumulate structures known from signal processing and instruction-set extensions allow to implement multiple-precision arithmetic at minimal hardware costs.

Fig. 3 shows the architecture of a GPS-enhanced RFID tag that uses a digit-level arithmetic unit to implement multiply-and-accumulate operations. There are two obvious reasons why a digit-level approach decreases the circuit's size: First the arithmetic unit itself is much smaller because its data width is only that of one digit (8 to 64 bits), and not of full precision. Second, the operands and temporary results can be stored in a hard-macro RAM, which is more area efficient than flip-flop-based storage. Most hard-macro RAM circuits are offered either as single-port or dual-port memories. The little area expense of the second port leads to great time savings in our application. Thus, a dual-ported RAM is used. One port reads a digit as input for the arithmetic unit while the other port is used to write back computed results at a different address.

Non-volatile memory is also necessary, to store domain parameters (e.g. secret key, modulus) and coupons. A *digital controller* implements the necessary algorithms to perform the overall computation, by means of digit-level operations which are carried out by the *arithmetic unit*. The arithmetic unit contains the actual datapath for performing the required operations. §4 details the arithmetic unit. The width of the data buses shown in Fig. 3 is equal to the digit size of the arithmetic unit. Thus one digit can be loaded from and stored to the RAM in every cycle. The digit size is a parameter fixed at synthesis time. Reasonable sizes are between 8 and 64 bits. Larger digit sizes seem impractical because the corresponding $d \times d$-digit multiplier would be too large. Digit sizes below 8 bits also seem impractical since that would cause too long computations. Computation time scales quadratically $\mathcal{O}(k^2)$ with the number of digits $k = \frac{n}{d}$ to represent a multi-precision word.

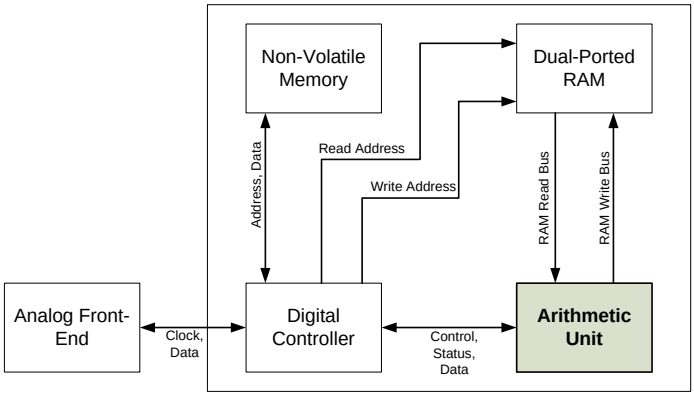

Fig. 3. Overview of an RFID tag, enhanced with a digit-level GPS architecture.

4 Arithmetic Unit

The GPS arithmetic unit must be capable of performing modular exponentiation ($x = g^r \bmod n$) for the coupon recalculation, and (non-modular integer) addition and multiplication ($y = r + s \cdot c$) for the response calculation during protocol execution. Our approach computes modular exponentiation in a sequence of square-and-multiply operations, which are broken down to digit level. The implementation efficiency of (modular) multiplication is crucial for the GPS hardware. It determines the circuit size and its performance.

4.1 Digit-Level Montgomery Multiplication

In our approach modular multiplication is implemented by Montgomery multiplication. There are several ways to break down Montgomery multiplication to digit-level operations. Five different approaches have been analyzed by Koç et al. in [11]. They differ in how much primitive operations (digit additions, digit multiplications, memory read/write) and how much temporary memory they require. The one which seems most suitable for our work, due to its low memory requirements and low number of operations, is referred to by Koç et al. as *Coarsely integrated operand scanning* (CIOS). [11] explains its details very well. Basically the CIOS algorithm works as follows: The first digit of the first operand is multiplied with all digits of the second operand to obtain a partial product. After that an interleaved reduction step takes place by adding a multiple of the modulus to the partial product to make it divisible by 2^d, and subsequently shifting the intermediate result one digit to the right (= division by 2^d). Then the next accumulation step is executed: The next partial product (obtained by multiplying the second digit of the first operand with all digits of the second operand) is added to the intermediate result. This is followed by

another interleaved reduction step. These accumulation and reduction steps are repeated for all remaining digits of the first operand.

4.2 Analyzing Digit-Level Operations

To perform the desired operations the arithmetic unit consists of two (or more) *input registers* (of size d) for the operands, and two *output registers* (of size d). Two output registers are necessary because digit-level operations can produce results which are larger than the digit size; two output registers are also sufficient, since no atomic digit-level operation of the CIOS algorithm from [11] can produce results larger than $2^{2d} - 1$. No extra carry registers are needed. The dominant operation performed during one GPS protocol execution is modular exponentiation, which is broken down into multiplications (by means of square-and-multiply). Single multiplications are performed using the CIOS algorithm. The dominant operation in this algorithm lies within the accumulation step. This operation calculates the sum of three terms: The current value of the `Output High` register, one digit from RAM, and the product of two more digits from RAM; i. e. the result is $D_1 + D_2 \cdot D_3 + H$, where D_i denote digits from RAM and H denotes the value of the `Output High` register at the time before the operation starts. All other (digit-level) operations (within the CIOS algorithm and within algorithms for multi-precision addition and multiplication) can be reduced to this operation. E.g. calculating just the product of two digits can be achieved by setting D_1 and H to 0. Our proposed arithmetic unit is designed to compute the operation $D_1 + D_2 \cdot D_3 + H$ most efficiently.

4.3 Schematic

The operation described above has three external inputs D_1, D_2, D_3, thus the most obvious architecture for an arithmetic unit has three input registers. For such a unit it would be possible to wire the arithmetic components (multiplier, adder) in a way that the operation could be executed within one single cycle. That is of course assuming that the input registers already hold the values D_1, D_2, D_3. Since only one digit per cycle can be loaded from a RAM hard-macro, spending the area for three input registers does not bring real speed-up. Thus it is reasonable to use only two input registers.

The schematic of the arithmetic unit is depicted in Fig. 4. A high-level model written in Java proved the efficiency of the approach. An implementation in Verilog is used for synthesis. Both the high-level model and the Verilog model are parameterizable concerning the digit size d. The $d \times d$-digit multiplier is implemented as pure combinational logic; its area demand scales quadratically with the digit size d. However, since only small digit sizes will be considered, this approach seems possible. Other multiplier architectures (e.g. bit-serial multiplication) would require additional resources (e.g. shift registers). They would also complicate control and prolong execution time. It should be noted that the implementation details of the multiplier have been left to the synthesis tool. A simple `assign c = a * b` statement was put into the Verilog code, so that the

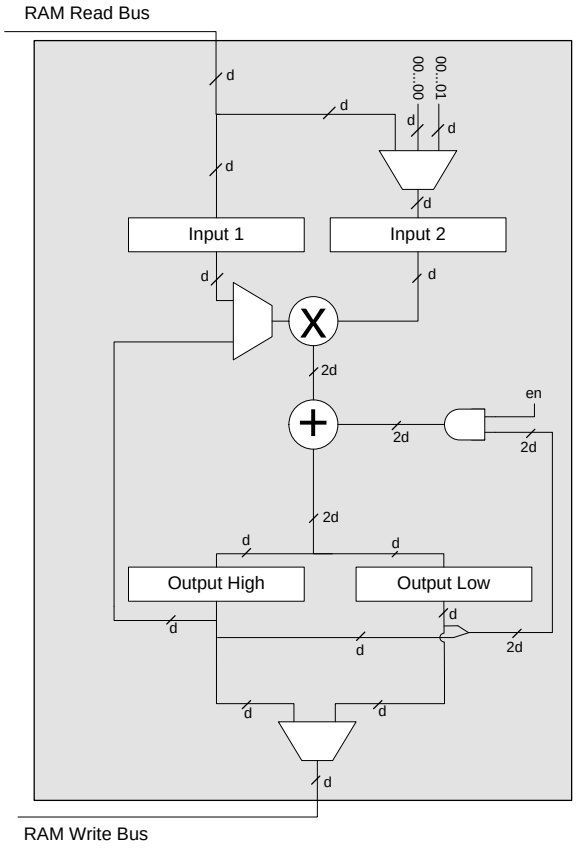

Fig. 4. Schematic of the arithmetic unit with digit size d.

synthesis tool would have the possibility to optimize the circuit while considering area and clock constraints. The input register D_2 has additional logic to be able to load either a value coming from the RAM read-bus (`din`), or one of the two constant values $(000\ldots00)_2 = 0$ and $(000\ldots01)_2 = 1$. The reason for this is explained in the next section. Although Fig. 4 shows a multiplexer, the most efficient implementation which achieves this behavior consists of d AND gates and one OR gate. Another possibility would be to store the constants $(000\ldots00)_2 = 0$ and $(000\ldots01)_2 = 1$ in memory and load them from there. This approach does not need additional logic in the arithmetic unit but the minimum size of the memory is increased by two entries. Furthermore the execution time of the overall computation will be increased with this approach because loading constant values into the second input register allocates the read bus of the memory. Meanwhile, no meaningful operation of the datapath is possible.

Table 1. Comparison of synthesis results of the arithmetic unit (not including RAM) for different CMOS technologies, and two different digit sizes.

Digit Size: 32 bits
(RAM for 151 digits needed, approx. 4.7 million clock cycles necessary for computing one coupon)

Technology	Area $[\mu m^2]$	Critical Path $[ns]$
AMS 0.35 [2]	361 561.20	28.93
UMC 0.25 [18]	181 470.96	20.94
UMC 0.13 [17]	44 115.45	10.86

Digit Size: 8 bits
(RAM for 560 digits needed, approx. 66.6 million clock cycles necessary for computing one coupon)

Technology	Area $[\mu m^2]$	Critical Path $[ns]$
AMS 0.35	43 880.47	10.16
UMC 0.25	20 599.92	5.98
UMC 0.13	4 855.68	3.41

Let us now investigate how the arithmetic unit depicted in Fig. 4 can be used to compute the modular exponentiation $x = g^r \bmod n$. As it has been explained, exponentiation is broken down to modular multiplication by the square-and-multiply approach. These multiplications are in turn broken down to digit-level operations. The dominant digit-level operation is $D_1 + D_2 \cdot D_3 + H$, as described in §4.2, where D_i denote digits in RAM and H denotes the current value of the `Output High` Register. The sum $D_1 + H$ can be accumulated in two steps by means of the feedback loops shown in Fig. 4. While doing so, the `Input 1` register is loaded with the constant value $(000\ldots01)_2 = 1$. That way the multiplier output is 1 times its left input. Then in the third step the values D_2, D_3 are loaded to the input registers and the product $D_2 \cdot D_3$ is added to the intermediate result $D_1 + H$, thus resulting in the final result $D_1 + D_2 \cdot D_3 + H$. All other digit-level operations are performed in a similar way.

5 Results

Our implementation of an arithmetic unit for coupon recalculation of the GPS authentication scheme is parameterizable with respect to the digit size d. Smaller digit sizes lead to smaller arithmetic units, but the time necessary to complete one authentication increases heavily with smaller digit sizes. The number of necessary digit-level operations is roughly proportional to the square of the number of digits per operand. Therefore in our opinion digit sizes below 8 bit are unpractical, considering that the full-precision size of the operands is (at least) 1024 bits.

Table 1 shows the synthesis results for the implementation of the arithmetic unit. For synthesis the *PKS Shell* from Cadence was used. Three different CMOS technologies have been used: A 0.35 µm CMOS technology from austriamicrosystems [2], and a 0.25 µm and a 0.13 µm CMOS technology from UMC [17, 18]. The critical path of the circuit is very short, due to the small arithmetic components, which are only of digit size. That means that it could be clocked very fast; frequencies up to 290 MHz are possible with UMC 0.13 µm CMOS technology. This opens a wide window of possibilities for operation. When operated at

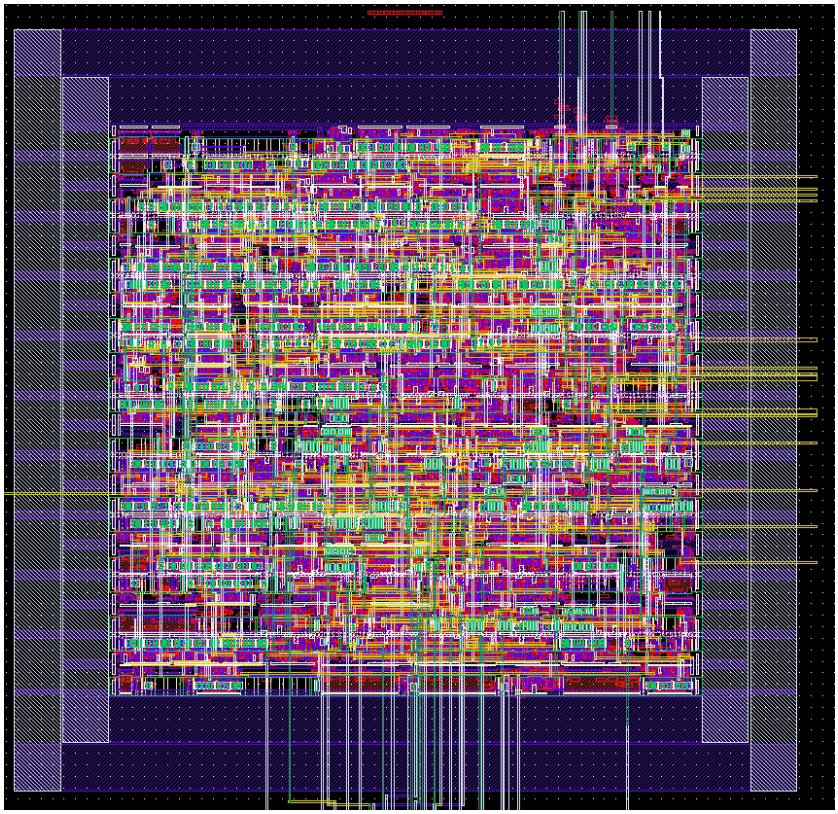

Fig. 5. Layout of the GPS arithmetic unit (digit size: 8 bits) after place-and-route.

very low clock frequencies (e.g. 100 kHz), the circuit consumes very little power (6.25 µW). We have performed a power simulation of the arithmetic unit (digit size: 8 bit), using the near-SPICE simulator *Nanosim* from Synopsys. The circuit was first synthesized for AMS 0.35 µm CMOS technology with *PKS Shell*, then placed and routed with *First Encounter*. The resulting layout can be seen in figure 5. A netlist for *Nanosim* was extracted with *Assura* from the layout after place-and-route. The simulation reveals that the arithmetic unit consumes approximately 2.5 µA when clocked with 100 kHz. Thus, with 2.5 V supply voltage, the power consumption is 6.25 µW. This is comparable to the results of Feldhofer et al. for AES [4]. They use the same CMOS technology and their AES implementation consumes 5 µW.

When raising the supply voltage to 3.3 V the performance of the circuit improves. It can be clocked with frequencies up to 125 MHz. At 125 MHz and 3.3 V supply voltage, the circuit draws a current of 3.2 mA. That means that if more power is available the computation can be sped up by a factor of 1250, compared to operation at 100 kHz.

The synthesis of the 8-bit arithmetic unit requires approximately 800 gate equivalents. After place-and-route the circuit takes up an area of approximately 64 250 µm^2. In addition, it would require a RAM hard-macro capable of storing 560 bytes. In comparison, the smallest implementation by McLoone et al. [12, 13] requires only 431 gate equivalents. However their implementation can only compute the response $y = r + s \cdot c$ in the last step of the GPS protocol, whereas our circuit is capable of calculating the modular exponentiation $x = g^r \bmod n$ to compute new coupons on-tag. Of course our implementation can also be used to calculate the response y. When using standard sizes for the parameters (cf. §2.1) this calculation takes 627 cycles, including time to load the operands to RAM. 33 of these cycles can be saved if a fresh coupon has been computed immediately before and thus the random value r is already present in RAM.

6 Conclusion

This paper introduced the coupon recalculation approach, which is an area- and power-efficient way to extend the *coupon approach* of the GPS authentication scheme. During the idle time of RFID tags, fresh coupons are computed for future authentication requests. A full-precision arithmetic unit would require approximately 50 000 gate equivalents, plus approximately 2200 bits of non-volatile memory to store keys and domain parameters (if typical values for GPS parameters are used). Our digit-level approach makes use of an arithmetic unit, which requires only approximately 800 GE for a digit size of 8 bits. In addition RAM resources for storing 560 bytes are needed. One coupon calculation takes about 66.6 million clock cycles, and the maximum clock frequency of the circuit on UMC 0.13 µm CMOS technology is approximately 290 MHz. The approach is very flexible and can be used either when very little power is available (i.e. low clock frequency, longer execution time). When more power is available, the computation can be accelerated by increasing the clock frequency. Up to four coupons can be computed per second under such conditions (290 MHz).

References

1. ADT Tyco Fire & Security, Alien Technology, Impinj Inc., Intel Corporation, Symbol Technologies, and Xterprise. RFID and UHF – A Prescription for RFID Success in the Pharmaceutical Industry. White paper, 2006.
2. austriamicrosystems. 0.35μm CMOS Process Standard-Cell Library. `http://asic.austriamicrosystems.com/databooks/index_c35.html`.
3. T. Blum and C. Paar. Montgomery modular exponentiation on reconfigurable hardware. In Koren and Kornerup, editors, *Proceedings of the 14th IEEE Symposium on Computer Arithmetic (Adelaide, Australia)*, pages 70–77, Los Alamitos, CA, 1999. IEEE Computer Society Press.
4. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen. AES implementation on a grain of sand. *IEEE Proceedings on Information Security*, 152(1):13–20, October 2005.

5. M. Girault. An identiy-based identification scheme based on discrete logarithms modulo a composite number. In I. Daamgard, editor, *Advances in Cryptology – Eurocrypt '90*, number 473 in Lecture Notes in Computer Science, pages 481 – 486. Springer, 1991.
6. M. Girault. Self-certified public keys. In D. Davies, editor, *Advances in Cryptology — Eurocrypt '91*, number 547 in Lecture Notes in Computer Science, pages 490 – 497. Springer, April 1992.
7. M. Girault. Low-size coupons for low-cost ic cards. In *Smart Card Research and Advanced Applications, Proceedings of the Fourth Working Conference on Smart Card Research and Advanced Applications, CARDIS 2000, September 20-22, 2000, Bristol, UK*, volume 180 of *IFIP Conference Proceedings*, pages 39–50. Kluwer, 2000. ISBN 0-7923-7953-5.
8. M. Girault and D. Lefranc. Public key authentication with one (online) single addition. In *Cryptographic Hardware and Embedded Systems — CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2004. ISBN 978-3-540-22666-6.
9. M. Girault, G. Poupard, and J. Stern. On the fly authentication and signature schemes based on groups of unknown order. *Journal of Cryptology*, 19(4):463–487, October 2006.
10. ISO/IEC. International Standard ISO/IEC 9798 Part 5: Mechanisms using zero-knowledge techniques, December 2004.
11. C. K. Koç, T. Acar, and B. J. Kaliski. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, June 1996.
12. M. McLoone and M. Robshaw. Public key cryptography and RFID tags. In *Topics in Cryptology — CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*. Springer, 2007. ISBN 978-3-540-69327-7.
13. M. McLoone and M. J. B. Robshaw. New architectures for low-cost public key cryptography on RFID tags. In *IEEE International Symposium on Circuits and Systems, ISCAS 2007*, pages 1827–1830, May 2007.
14. P. L. Montgomery. Modular multiplication without trial division. In *Mathematics of Computation*, volume 44, pages 519 – 521, 1985.
15. NESSIE. Final report of European project number IST-1999-12324, named new european schemes for signatures, integrity, and encryption. `https://www.cosic.esat.kuleuven.be/nessie/Bookv015.pdf`, April 2004.
16. P. Tuyls and L. Batina. RFID-Tags for Anti-counterfeiting. In D. Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer, 2006.
17. UMC. UMC standard cell library — 130 nm CMOS process.
18. UMC. UMC standard cell library — 250 nm CMOS process.
19. J. Wolkerstorfer. Is Elliptic-Curve Cryptography Suitable for Small Devices? In *Workshop on RFID and Lightweight Crypto, July 13-15, 2005, Graz, Austria*, pages 78–91, 2005.