Low-Cost SHA-1 Hash Function Architecture for RFID Tags

M. O'Neill (nee McLoone)

Institute of Electronics, Communications,
and Information Technology (ECIT)
Queen's University Belfast, Northern Ireland
m.oneill@ecit.qub.ac.uk

Abstract. Radio frequency identification (RFID) tags are expected to play an important role in the future of ubiquitous computing. By extending the deployment of computing devices to every facet of our lives will open up communication infrastructures to new forms of attack. As such, the inclusion of security is vital in the development of next-generation wireless and ubiquitous devices. In this paper, a low-cost SHA-1 hash function architecture is described that can be used to provide data integrity and authentication in RFID tags. When implemented on 130 nm CMOS the design utilises 5527 gates and consumes 2.32 μW of power. To the author's knowledge, the proposed architecture is the smallest and most power-efficient hash function design reported in the literature to date.

1 Introduction

Dramatic advances in digital wireless technology over the past two decades have led to many exciting developments including the rapid growth of mobile and ubiquitous computing. In the future, through the use of mobile applications and devices embedded in the surrounding environment, users will be offered transparent computing and communication services at all times and in all places. Applications of wireless mobile and ubiquitous computing, such as smart homes, smart automobiles, and remote payment have already begun to emerge. Security is an important factor that must be taken into consideration if the uptake of this new computing paradigm is to be successful. The capability of digital devices to autonomously interact brings with it significant security and privacy risks for the end user [1, 2].

Radio frequency identification (RFID) tags will play a key role in the future development of ubiquitous computing. The current deployment of low-cost RFID tags in applications such as access control, inventory control, luggage tracking and product traceability has already gained much research and media attention. However, the fact that RFID systems involve contactless communication, are non line-of-sight and that tags broadcast information, raises significant security concerns including violation of privacy, consumer tracking and product forgery. The draft recommendation on RFID privacy and security published by the European Commission in February 2008 stated that RFID applications need to operate in a secure manner and that research needs to be carried out into high-performance and low-cost security solutions for RFID devices [3].

RFID tags can be active, i.e. self-powered, or passive, where they are powered by the electromagnetic field of a reader. The cheaper devices will typically be of the second kind and in this research an RFID system is assumed to consist of a more powerful reader and a set of low-cost tags. The reader consists of a transmitter and receiver, control functionality and a coupling element to interact with the tag. An RFID tag comprises a coupling element with an antenna, and a chip that gives the tag some limited functionality [4]. It is clear that device costs will be a significant factor in the development of ubiquitous computing, just as for current RFID deployment today. Devices must be cheap enough for an application to be financially viable and so while the universal deployment of RFID tags is acknowledged to bring security and privacy threats, any security countermeasures should incur minimal additional cost. Due to a tag's limited resources, the addition of security measures and techniques poses very interesting challenges for the research community. It is important to consider the cost of including security architectures in tags and three factors which need to be analysed are peak power consumption, computation time and silicon requirements. In current debates about RFID deployment, estimates vary on the space that might be available in the more restricted devices. The measure of gate-equivalents (GE) allows a technically-neutral estimate of the physical space required and within the cryptographic community there is an oft-quoted consensus that out of 1000 – 10,000 GE on a restricted device, around 200 – 3000 GE might be available for security [5, 6]. If Moore's Law [7] is applied, then it may be possible to expect around 10,000 GE to be available for security features at a similar price in four to five years.

Hash functions can be used to provide data integrity and, in conjunction with digital signature algorithms and message authentication codes, to provide authentication. A security level of 80 bits is often deemed adequate for RFID tag applications. This was taken as the security level requirement in the eSTREAM project [8] in the selection of stream cipher candidates suitable for low-resource hardware implementation. Therefore, a hash function with an output $\geq$ 160 bits is required in order to provide RFID tag security. Hence, a low-cost SHA-1 architecture, which has a 160-bit hash output, is proposed in this research. Weaknesses have been discovered in the SHA-1 hash function, which show that only 2^{69} operations are required in order to find a collision [9], which is much less than the 2^{80} operations required using a brute force attack. However, the security requirements of certain RFID applications may not require collision resistance.

Previous research has been carried out on low-cost SHA-1 hash function architectures for ubiquitous and RFID applications. Feldhofer and Rechberger [10] presented low-power architectures of the SHA-1 and SHA-256 hash functions for RFID protocols. Their SHA-1 design required 8120 gates and consumed 35.24 μW (10.68 μA) of power. A low-power SHA-1 design described by Kaps and Sunar [11] targeted ubiquitous computing. However, it was not a full implementation as they assumed that values required in the algorithm's message schedule were stored in external memory. Satoh and Inoue's low-area SHA-1 architecture [12] required 7971 gates, but power measurements were not provided. Finally, Choi *et al.* [13] outlined a low-power SHA-1 implementation for RFID systems that utilised 10,641 gates and consumed 19.5 μW of power. However, the area of all these previous designs is still too large for current RFID tag deployment. Low-cost designs of digital signature algorithms that incorporate the SHA-1 hash function have also been investigated for RFID applications. This research has involved elliptic curve cryptography (ECC) based architectures such as the work by Fürbass and Wolkerstorfer [14] on a low-cost Elliptic Curve Digital Signature Algorithm (ECDSA) design over GF(P^{192}) and the research by Schroeppel *et al.* [15] describing a hardware architecture of the Elliptic Curve Optimal El Gamal Signature scheme over GF(2^{178}). Many new security protocols being proposed for RFID tags also require hash functions [16–18]. Therefore, it is evident that research is required into the design of new low-cost hash function algorithms or into developing highly optimised architectures of existing hash functions, which is the focus of this work.

All of the previous research into low-cost SHA-1 designs have employed a 32-bit architecture since the SHA-1 algorithm comprises inherently 32-bit operations. In order to achieve a highly optimised SHA-1 design, an 8-bit hardware architecture is proposed in this research. A hardware architecture is considered since hardware offers real-time security, lower power and is inherently more tamper-proof than software. These advantages are vital if security is to be provided in next-generation wireless applications so that end-users are provided with effective security with little or no overhead cost. Also, the performance analysis of hardware-based security designs is essential to ascertain the true suitability and practicality of such schemes in resource constrained RFID applications.

The SHA-1 hash function is outlined in section 2 of this paper. The proposed 8-bit low-cost SHA-1 hardware architecture is described in detail in section 3. A performance analysis and comparison with previous work is provided in section 4 and conclusions are discussed in section 5.

2 SHA-1 Hash Function

The Secure Hash Algorithm (SHA-1) was proposed by the US National Institute of Standards and Technology (NIST) in 1995 [20]. It operates on a message of length $< 2^{64}$ in 512-bit blocks and cycles through 80 iterations of a hash computation to produce a 160-bit message digest. The algorithm comprises three principle steps: Message Pre-processing, the Message Schedule and the Hash Computation.

The message pre-processing stage involves padding the message to a length $\equiv 448 \texttt{ mod } 512$, appending the message length as a 64-bit number and parsing the padded message into N 512-bit data blocks. The message schedule involves the generation of 80 32-bit values, W_t, which are utilised in each hash computation iteration, and calculated as,

$$W_t = \begin{cases} Message_t & 0 \leq t \leq 15 \\ ROT_{LEFT_1}(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \leq t \leq 79 \end{cases} \quad (1)$$

where $ROT_{LEFT_n}(word)$ is a circular rotation of a word by n positions to the left.

Finally, the hash computation involves the update of 5 variables, *a, b, c, d* and *e*, every iteration according to equation 2, where the initial values for these variables, as given in equation 3, are provided in the original SHA-1 algorithm specification [20]. The function, $F_t(b, c, d)$, is described in equation 4 and the 32-bit constant values, K_t are given in equation 5. After 80 iterations the 160-bit output is added to the initial a to e values, and the result initialises the a to e values for the next data block to be processed. After all N data blocks have been processed, the final output forms the 160-bit message digest. An outline of the hash computation is shown in Figure 1.

$$\begin{aligned} &T = ROT_{LEFT_5}(a) + F_t(b,c,d) + e + K_t + W_t \\ &e = d \\ &d = c \\ &c = ROT_{LEFT_{30}}(b) \\ &b = a \\ &a = T \end{aligned} \quad (2)$$

$$\begin{aligned} A &= 67452301 \\ B &= efcdab89 \\ C &= 98badcfe \\ D &= 10325476 \\ E &= c3d2e1f0 \end{aligned} \quad (3)$$

$$F_t(b,c,d) = \begin{cases} (b\ AND\ c)\ OR\ (\bar{b}\ AND\ d) & 0 \leq t \leq 19 \\ b \oplus\ c \oplus\ d & 20 \leq t \leq 39 \\ (b\ AND\ c)\ OR\ (b\ AND\ d)\ OR\ (c\ AND\ d) & 40 \leq t \leq 59 \\ b \oplus\ c \oplus\ d & 60 \leq t \leq 79 \end{cases} \quad (4)$$

$$\begin{aligned} K_t &= 5a827999 & 0 \leq t \leq 19 \\ K_t &= 6ed9eba1 & 20 \leq t \leq 39 \\ K_t &= 8f1bbcdc & 40 \leq t \leq 59 \\ K_t &= ca62c1d6 & 60 \leq t \leq 79 \end{aligned} \quad (5)$$

3 Low-Cost 8-bit SHA-1 Hardware Architecture

The SHA-1 algorithm is intrinsically designed to be implemented on a 32-bit platform - the logical function, F_t, and rotate functions operate on 32-bit words and the addition is performed modulo 2^{32}. As such, all implementations of SHA-1

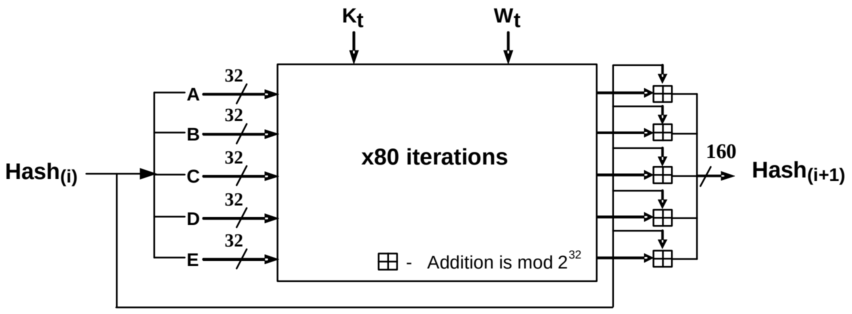

Fig. 1. Outline of Hash Computation in SHA-1

to date have been based on a 32-bit architecture. Since the goal of this research is to design a low-cost SHA-1 architecture, an 8-bit design is considered, where all of the 32-bit oriented operations in the algorithm are modified such that they can be performed in 8-bit blocks. The pre-processing step is assumed to be carried out in software. In our 8-bit SHA-1 design, the message blocks, $Message_t$, are loaded into the design in 64 x 8-bit blocks and the message schedule is designed using a 64 x 8-bit shift register array, as shown in Figure 2. As such, the original message schedule can be rewritten as:

$$W_t = \begin{cases} Message_t & 0 \le t \le 63 \\ ROT_{LEFT_1}(W_{t-12} \oplus W_{t-32} \oplus W_{t-56} \oplus W_{t-64}) & 64 \le t \le 319 \end{cases} \quad (6)$$

The SHA-1 message schedule involves 32-bit XOR and rotate functions. The 32-bit XOR functions can be easily broken down into 8-bit XORs carried out over 4 clock cycles to provide the same overall result. However, when performing the inherently 32-bit rotate function, $ROT_{LEFT_1}(x)$, in 8-bit blocks, the rotated bit must be taken into account every 4th data block as the newly generated W_t value is stored back into the register array. As such, some additional control logic is required. The overall 8-bit SHA-1 message schedule is outlined in Figure 2. An example illustrating how the rotation of a 32-bit data block, x, is carried out in 8-bit blocks is given in Figure 3, where,

$$x = \begin{array}{cccc} \texttt{11100001} & \texttt{01100010} & \texttt{01100011} & \texttt{01100100} \\ \textit{x3} & \textit{x2} & \textit{x1} & \textit{x0} \end{array}$$

In cycle 1, the seven least significant bits (LSBs) of *x0* are concatenated with the output of register A (see Figure 2 – initialised to $'0'$) and input into register W_{63}. The most significant bit (MSB) of *x0* is input into register A. In cycle 2 the seven LSBs of *x1* are concatenated with the output of register A (now contains MSB of *x0*) and input into W_{63}. The MSB of *x1* is then input into register A. The process is similar in cycle 3 for *x3*. The procedure differs in cycle 4 in that now the MSB of *x4* is not input into register A but instead forms the LSB of *x0*

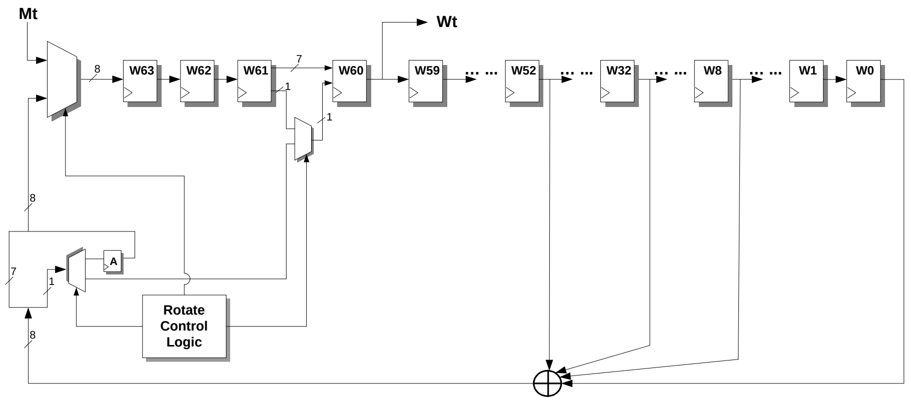

Fig. 2. 8-bit SHA-1 Message Schedule

	Input	*W63*	*W62*	*W61*	*W60*/Output
Cycle 1	*x*0: 0110 0100	1100 1000	1100 1000	1100 1000	1100 1001
Cycle 2	*x*1: 0110 0011	1100 0110	1100 0110	1100 0110	1100 0110
Cycle 3	*x*2: 0110 0010	1100 0100	1100 0100	1100 0100	1100 0100
Cycle 4	*x*3: 1110 0001	1100 0010	1100 0010	1100 0010	1100 0010

Fig. 3. ROT_{LEFT_1} of a 32-bit Data Block carried out in 8-bit blocks

as it is input into register W_{60}. Thus, after 4 cycles the result that would have been achieved from the 32-bit rotate function is now obtained using 8-bit data blocks.

In the 8-bit hash computation architecture, the 32-bit variables a to e are considered as 8-bit data blocks, *a0*, *a1*, *a2*, *a3* to *e0*, *e1*, *e2*, *e3*. Therefore, the architecture is designed using a 20 x 8-bit shift register array, as illustrated in Figure 4. Within this step, the operations that need to be carefully considered when employing an 8-bit architecture are the calculation of c, which involves an inherently 32-bit rotate function, $ROT_{LEFT_{30}}(b)$, and the calculation of a, which consists of a rotate function, $ROT_{LEFT_5}(a)$, a logic function and addition modulo 2^{32}. The $ROT_{LEFT_{30}}(b)$ function can be taken to be equivalent to a $ROT_{RIGHT_2}(b)$ function. The function takes 4 cycles to complete when performed in 8-bit blocks and the two rotated bits are taken into account every 4th cycle. In cycle 1, the two LSBs of *b3* are input into registers and stored for 3 cycles, as shown in Figure 4. For cycles 1, 2 and 3 the two LSBs of *b2* are concatenated with the six MSBs of *b3* to form the input into *c0*. On the 4th cycle the last byte of the rotated result is determined by concatenating the output of the registers with the six MSBs of *b3*.

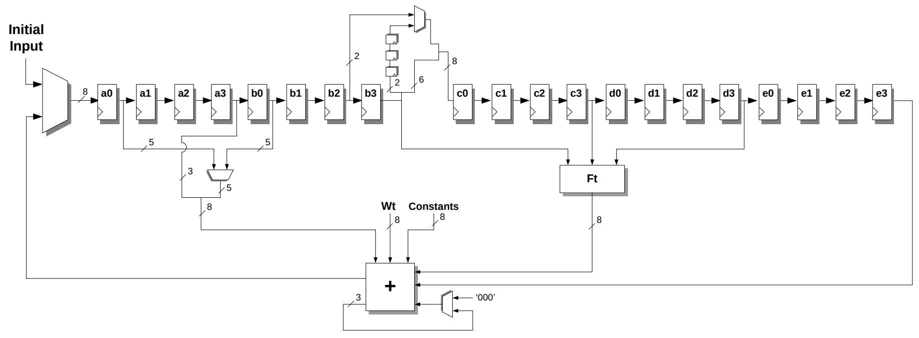

Fig. 4. 8-bit Hash Computation Architecture

Since the logic function, $F_t(b, c, d)$, comprises XOR, AND, OR and NOT operations, reducing it to operate on 8-bit blocks over 4 clock cycles will have no affect on achieving the equivalent 32-bit result. The $ROT_{LEFT_5}(a)$ function required in the calculation of a is also carried out over 4 cycles. The design of this function does not require any additional registers as the values used to perform the rotation can be read from existing registers. In cycle 1, the first byte of the rotated result can be formed by concatenating the three LSBs of *a3* with the five MSBs of *a0*. Then in cycles 2, 3 and 4 the remaining bytes are determined by concatenating the three LSBs of *a3* with the five MSBs of *b0*, since the contents of register *a3* will pass to register *b0* every cycle. When performing the addition required in the calculation of a in 8-bit blocks, an appropriate carry has to be included to ensure that the result matches that obtained from an equivalent 32-bit addition.

The 8-bit hash computation design will take 320 cycles to complete. The result must be added to the initial a to e values to form the new a to e values utilised in processing the next data block. As the initial 8-bit *a0*, *a1*, *a2*, *a3* to *e0*, *e1*, *e2*, *e3* values are input into the shift register array they are also stored in a memory (a register array) so that they are available for the final addition. In the final addition the output of register *e3* is added to the 8-bit register array output and once again, a carry is used to ensure that the final result corresponds to what would be obtained from the corresponding 32-bit addition. Since the output of the message schedule is not available for 4 cycles and the 160-bit hash result is output in 8-bit data blocks over 20 cycles, the overall 8-bit SHA-1 architecture requires 344 clock cycles to complete.

4 Performance Evaluation

The 8-bit SHA-1 architecture was implemented using the Faraday UMC 180 nm (L180_GII) and UMC 130 nm (L130_LL) CMOS libraries. It was tested using `Modelsim`, synthesised using `Synopsys Physical Compiler Version 2006.06` and its power consumption obtained from `Synopsys PrimeTime PX Version 2007.06`. The power consumption was calculated as:

$$P_{max} = P_{ave} + 2 * StdDev$$

for a set of randomly generated input values. A breakdown of the area utilised by each component of the proposed 8-bit low-cost SHA-1 design is given in Table 1. The overall performance results of the 8-bit design and previously proposed low-cost 32-bit SHA-1 implementations are provided in Table 2. The SHA-1 architecture by Satoh and Inoue [12] comprises four adders for the additions: $e + F_t(b, c, d) + W_t + K_t + ROT_{LEFT_5}(a)$. In order to reduce the overall design area they reuse the adder that is used to compute $e + F_t(b, c, d)$ to carry out the addition between the final a to e values and the initial a to e values over 5 cycles. They perform one SHA-1 round in 1 clock cycle and therefore, their design requires 85 clock cycles to complete. No power measurements are provided for their proposed architecture. The design by Kaps and Sunar [11] is not a full implementation of SHA-1 as they use external memory to store the W_t values generated in the message schedule, which would not be feasible if employed within RFID tags. The registers used to store the W_t values in the proposed 8-bit SHA-1 design implemented on 130 nm technology account for 2560 gates and 46% of the overall design. In Kaps and Sunar's implementation of this operation, they read the four 32-bit values, W_{t-3}, W_{t-8}, W_{t-14} and W_{t-16}, from the external memory and write the result, W_t, back into memory in each round. Therefore, their design requires 5 cycles to compute any one SHA-1 round. Since they also perform the final addition over 5 cycles, the total number of clock cycles for their low-cost SHA-1 design is 405. The SHA-1 architecture by Feldhofer and Rechberger [10] is designed such as only one 32-bit word is clocked in any one clock cycle. Their design requires 1724 clock cycles to complete. Choi *et al.* [13] use one adder to perform the hash computation operation in their SHA-1 architecture and as such, one SHA-1 round function is achieved in 4 clock cycles. They utilise a further 10 clock cycles, possibly to perform initialisation and the final addition. Overall, they require 330 clock cycles to complete a full SHA-1 operation. As described in Section 3, the 8-bit SHA-1 architecture outlined in this paper takes 344 clock cycles to complete, where one SHA-1 round requires 4 clock cycles since operations are carried out in 8-bit data blocks.

It is very difficult to compare power consumption across different technologies. For example, the power consumption associated with the Faraday UMC 180 nm and the UMC 130 nm libraries is 29 nW/MHz/gate and 6 nW/MHz/gate respectively. However, whatever the target technology it is important that the power consumption of the design meets the limitations imposed by RFID tags.

Table 1. Area Utilised by Components in 8-bit SHA-1 Architecture

Component	Area (0.13 μm) (gates)	Area (0.18 μm) (gates)
Hash Computation	2751	3160
Message Schedule	2655	2831
Control logic	121	131
Total	5527	6122

According to Feldhofer *et al.* [21], the current consumption of a security architecture for implementation on RFID tags must not exceed 15 μA. For 1.3 V and 1.8 V CMOS technologies, this is equivalent to 18 μW and 27 μW respectively. The proposed 8-bit SHA-1 design meets these power constraints comfortably. Gate count can be used to provide an approximate comparison of area across different technologies (although it can vary by $\approx$ 10% – note the difference in gate count for the proposed design in 130 nm and 180 nm technologies). In RFID tags the silicon area requirement significantly impacts the cost and the cost per mm^2 of silicon is estimated at 4 cent [22]. Therefore, it is vital that the silicon area overhead resulting from the inclusion of security on low-cost tags is kept to a minimum. From Table 2, it is clear that the proposed 8-bit SHA-1 architecture is the smallest full SHA-1 design reported in the literature and is within close reach of current RFID tag deployment. This is achieved by using an 8-bit data-path which reduces the 32-bit XOR, AND, NOT and OR functions to equivalent 8-bit functions with no logic overhead and the 32-bit addition modulo 2^{32} and rotate operations to equivalent 8-bit operations with minimal control logic overhead. Using this design methodology the overall saving in area is approximately 1200 gates. In relation to timing, a tag must respond to a reader's request within 32 μs in accordance with the ISO/IEC 18000 standard [23]. An interleaved challenge-response protocol could be used such as that proposed by Feldhofer *et al.* [21], in which the response time is 18 ms. This corresponds to 1800 clock cycles when operating with a clock frequency of 100 kHz. The proposed design also satisfies this requirement.

5 Conclusions

Hash functions currently play an important role in providing data security to communication applications and they will continue to be required in the provision of security in next-generation wireless and ubiquitous devices. Although the SHA-1 algorithm is considered weak in comparison to other hash functions, it will remain suitable for some RFID applications in which collision resistance may not be essential. In this paper, a low-cost SHA-1 architecture that is based on an 8-bit datapath is presented which results in a significant reduction in area ($\approx$ 1200 gates) over previous work. Overall, the architecture is within close reach

Table 2. Performance Comparison of Low-cost SHA-1 Hardware Implementations

Design	Area (gates)	Power Consumpt. (μW@100 kHz)	Timing (clk cycles)
This work: 8-bit SHA-1 (0.13 μm/1.2 V)	5527	2.32	344
This work: 8-bit SHA-1 (0.18 μm/1.8 V)	6122	13.8	344
Feldhofer & Rechberger [10] (0.35 μm/3.3 V)	8120	35.24	1274
Kaps & Sunar [11] (0.13 μm/1.2 V)	4276 *partial design*	26.73 @500 kHz	405
Choi *et al.* [13] (0.25 μm)	10641	19.5	330
Satoh & Inoue [12] (0.13 μm)	7971	-	85

of current RFID tag deployment and will certainly be feasible for providing security in tags in the very near future. Future work will investigate a low-cost design of the more secure SHA-256 hash function using the 8-bit design methodology described in this paper.

References

1. F. Stanjano. The Resurrecting Duckling: security issues for ubiquitous computing, IEEE Computer, Vol.35(4), 2002.
2. J.I. Hong. Minimizing Security Risks in Ubicomp Systems," Computer, vol. 38, no. 12, pp. 118-119, Dec., 2005.
3. European Commission, Draft Recommendation on the implementation of privacy, data protection and information security principles in applications supported by Radio Frequency Identification (RFID), `http://ec.europa.eu/yourvoice/ipm/forms/dispatch?form=RFIDRec`.
4. K. Finkenzeller. RFID Handbook. John Wiley, 2003.
5. A. Juels and S. Weis. Authenticating Pervasive Devices with Human Protocols. In V. Shoup, editor, Advances in Cryptology - Crypto 05, LNCS 3126, 293-198, Springer- Verlag, 2005.
6. P. Peris-Lopez, J.C. Hernandez-Castro, J.M. Estevez-Tapiador, A. Ribagorda, 'RFID Systems: A Survey on Security Threats and Proposed Solutions', International Conference on Personal Wireless Communications - PWCA'06, LNCS 4217, 2006
7. G.E. Moore. Cramming More Components Onto Integrated Circuits. Electronics, April 19, 1965.`http://www.intel.com`.
8. ECRYPT, The eSTREAM Project, `http://www.ecrypt.eu.org/stream/`
9. X. Wang, Y.L. Yin, and H. Yu, Finding collisions in the full SHA-1, Advances in Cryptology CRYPTO 2005, LNCS 3621, pp.1736, Springer-Verlag, 2005.

10. M. Feldhofer, C. Rechberger, A Case Against Currently Used Hash Functions in RFID Protocols, First International Workshop on Information Security (IS'06), LNCS Vol. 4277, Springer, pp. 372-381, Montpellier, France, 2006.
11. J.P. Kaps,B. Sunar, Energy Comparison of AES and SHA-1 for Ubiquitous Computing, Embedded and Ubiquitous Computing (EUC-06) Workshop Proceedings. In Xiaobo Zhou et al. (Eds.), LNCS, Springer, 2006.
12. A. Satoh, T. Inoue, ASIC-Hardware-Focused Comparison for Hash Functions MD5, RIPEMD-160 and SHS, Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05), pp. 532-537, April 2005.
13. Y. Choi,M. Kim, T. Kim, H. Kim, Low Power Implementation of SHA-1 Algorithm for RFID System, IEEE 10th International Symposium on Consumer Electronics (ISCE'06), pp.1-5, 2006.
14. F. Fürbass, J. Wolkerstofer, ECC Processor with Low Die Size for RFID Applications, IEEE International Symposium on Circuits and Systems (ISCAS 2007), pp. 1835-1838, May 2007.
15. R. Schroeppel, C. Beaver, R. Gonzales, R. Miller, T. Draelos, A Low-Power Design for Elliptic Curve Digital Signature Chip, Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), LNCS 2523, pp. 366-380, 2002.
16. T. Dimitriou. A Lightweight RFID Protocol to protect against Traceability and Cloning attacks. First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm 2005), pp.59-66, Greece, IEEE Computer Society, 2005.
17. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic Approach to Privacy-friendly tags. In RFID Privacy Workshop, MIT, USA, 2003.
18. G. Avoine, P. Oechslin. A Scalable and Provably Secure Hash-Based RFID Protocol.The 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005), Hawaii, pp.110-114, IEEE Computer Society Press, 2005.
19. E. Asanghanwa, Using RFID Technology to Stop Counterfeiting', Whitepaper, Atmel Corporation, RSAConference365, `http://www.rsaconference.com/Security_Topics/Deployment_Strategies.aspx`
20. Secure Hash Standard, FIPS PUB180-1, US National Institute of Standards and Technology, April 1995.
21. M. Feldhofer, S. Dominikus, J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In M. Joye and J.-J. Quisquater, editors, Proceedings of CHES 2004, LNCS 3156, pp. 357-370, Springer Verlag, 2004.
22. S.E. Sarma, Towards the 5 tag, Technical report, MIT-AUTOID-WH-006, MIT-AUTOID Center, February 2001.
23. International Organisation for Standardisation. ISO/IEC 18000-3. Information Technology AIDC Techniques - RFID for Item Management, March 2003.