Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 115–122

Procedia
Computer Science

The 2nd International Conference on Ambient Systems, Networks and Technologies

A Secure Platform of Wireless Sensor Networks

Abidalrahman Moh'd[a], Hosein Marzi[b], Nauman Aslam[a], William Phillips[a], William Robertson[a], a*

[a]Department of Engineering Mathematics and Internetworking, Dalhousie University
1340 Barrington Street, Halifax, Nova Scotia, B3J 2X4, Canada
[b]Department of Information systems, St. Francis Xavier University
Antigonish, Nova Scotia, B2G 2W5, Canada

Abstract

Security was not considered when current wireless sensor nodes were designed. As a result providing high level of security on current WSNs platforms is unattainable, especially against attacks based on key resolving and node compromise. In this paper we scrutinize the security holes in current WSNs platforms, and compare the main approaches to implementing the cryptographic primitives used to provide security services for these platforms, in terms of security, energy, and time efficiency. To secure these holes and provide more efficiency we propose a custom hardware platform for WSNs. The choice of cryptographic primitives for our suggested platform is based on their compatibility with the constrained nature of WSNs and their security status. We also discuss the most efficient configurations and implementation methodologies of these primitives, and review their specialized implementations for WSNs in recent literature. Based on that, we provide a hardware implementation of a crypto-processor using Very high speed integrated circuit Hardware Description Language (VHDL). Experimental results using synthesis for Spartan-6 low-power FPGA shows that the proposed protocol outperforms related work in terms of computation time and energy consumption.

Keywords: Wireless Sensor Networks; Security Trusted Platform; Advanced Encryption Standard; Elliptic Curve Cryptography; Secure Hashing Functions

1. Introduction

The future of wireless sensor networks (WSNs) is promising; they are being deployed in many real-world applications, in the context of Ubiquitous Computing, Pervasive Computing, and Ambient Intelligence. It is expected that new WSNs applications will have greater share of the embedded systems market, with a growth rate of up to 50% per year [1]. These types of applications force the need of securing WSNs, which is very difficult to realize, due to scarceness of energy, constrained nature of the sensor platforms, openness of wireless communication, and the intensive mathematical computations of cryptographic primitives. Because of these unique challenges, the security of these networks became a very hot research field.

Many researchers proposed mechanisms to provide security for WSNs in literature. Regardless of the efficiency of these mechanisms, they will not provide sufficient security if they are implemented on top of unsecure platforms.

* Corresponding author: Abidalrahman Moh'd. Tel.: +1-902-402-3210; fax: +1-902-423-1801.
E-mail address: Abidalrahman.Mohd@dal.ca

doi:10.1016/j.procs.2011.07.017

The current WSNs nodes are being made using "off the shelf" components designed for other environments. Secure design for unique WSNs environment was not considered when they were manufactured.

Generally, three types of cryptographic primitives are used to provide security services, public key primitives, private key primitives, and hashing functions. Efficient implementations of these primitives for WSNs have been addressed by many researchers in the literature. These implementations can be done either in software or hardware. In this paper we review both approaches and discuss their security and efficiency. We also propose a custom hardware platform that guarantees maximum security and energy efficiency. Being closed and fully separated from operating system helps to avoid the threats and security holes in the operating system security. In addition, implementing it in hardware is more secure and energy efficient. The choice of cryptographic primitives for our suggested platform is based on their compatibility with the constrained nature of WSNs and their security status. The work in this paper can be seen as an initial step in redesigning the wireless sensor node from scratch with security as a main design goal.

The rest of the paper is organized as follows; in Section 2, we review previous approaches to WSNs security, their advantages, disadvantages, and security holes. Our proposed custom hardware platform for WSNs is presented in Section 3. In Section 4, we overview the most appropriate cryptographic primitives for WSNs, discuss their security status, and review their specialized implementations for WSNs in recent literature. Experimental energy and time results for implementing the best architectures for the selected primitives in hardware are provided in section 5. Finally, the conclusion and directions of our future work are presented in Section 6.

2. Approaches to WSNs Security

To implement cryptographic algorithms researchers followed two approaches, software implementations, and hardware implementations. The main advantage of software implementations is their flexibility, because changing them doesn't require any modification in the hardware architecture of wireless sensor nodes. However, these implementations have large processing overhead on weak WSNs processors. Table 1 shows the energy consumption and execution time of public key software implementations on Mica2 wireless node [2]. The relatively high processing time could result in missing some events or not reporting them in timely manner. It also affects the execution of other programs running on the node, and most importantly it consumes a large amount of energy.

Table 1: Energy and time results of public key software implementations

Operation	Key generation	ECDSA signature	ECDSA verification	D-H key exchange	El-Gamal encryption	El-Gamal decryption
Time (s)	6.74	6.88	24.17	17.28	24.07	17.87
Energy (mJ)	101.1	103.2	362.6	259.2	361.1	268.1

Software implementations are not only less efficient in terms of energy and time, but also they are less secure. As Fig. 1(a) illustrates, the security of software implementations is dependent on the security of the operating system and the underlying wireless node hardware platform, both of them were not designed to be secure.

Because software implementations are run on top of the operating system and share memory space with other running programs, they are vulnerable in terms of ease of modification and compromising keys. A clear example of such attacks in literature is the Cache-Collision Timing Attacks [3] on software implementations of Advanced Encryption Standard (AES) algorithm. This attack is based on exploiting characteristics of AES table lookups. It requires direct observation of memory before and after encryption by running the attacking process and AES algorithm on the same operating system.

Compared to software implementations of encryption algorithms, hardware implementations are not only much faster, but they also consume less energy up to a factor of 10^{-3} [2]. Some researchers tried to use "off the shelf" hardware components to provide encryption for WSNs. like the use of Chipcon CC2420 transceiver chip in [4]. This chip includes hardware implementation of The AES algorithm, and the use of commodity Trusted Platform Module (TPM) chip for RSA public key encryption in [5]. In both cases, the communication between the security chip and other wireless node hardware components (i.e. the data bus between the TRM chip, controller and external memory) is done in plaintext. Spying on this communication will reveal valuable data such as the encryption keys.

Other attacks on hardware like cold boot attack [6] can also be used to get the content of the memory and resolve the encryption keys, because the keys are stored in plaintext. Software attacks through operating system are possible as well, because the keys are accessible by the operating system in plaintext. Fig. 1 (b) illustrates the security view of these implementations.

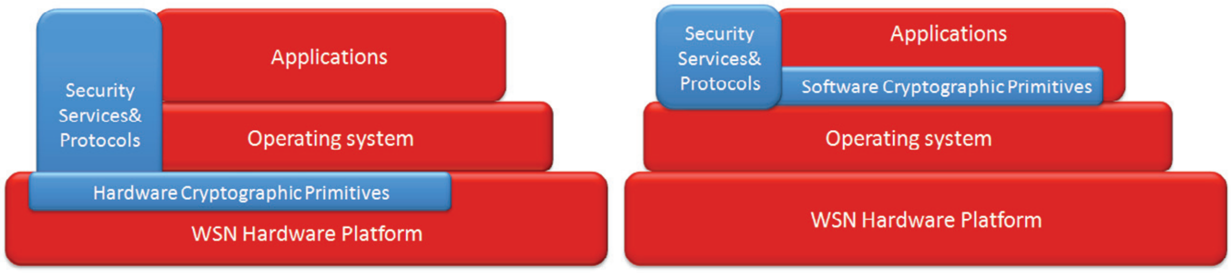
Security Services& Protocols
Applications
Operating system
Hardware Cryptographic Primitives
WSN Hardware Platform
Security Services& Protocols
Applications
Software Cryptographic Primitives
Operating system
WSN Hardware Platform

Fig. 1. (a) Software cryptography over unsecure platform and OS; (b) Hardware cryptography over unsecure platform and OS

3. A Custom Hardware Platform for WSNs.

In order to guarantee maximum security and energy efficiency for WSNs, we suggest a custom hardware platform that provides hardware-based key generation, storage, and encryption without any involvement from the operating system. Being closed and fully separated from operating system helps to avoid the threats and security holes in the operating system security. In addition, implementing it using hardware is more secure and energy efficient.
Our suggested hardware secure platform for WSNs is shown in Fig. 2. The only part of data that can be in plaintext is the data need to be processed inside the main processor, part of the internal memory, and the input data from sensors through the analog to digital converter (ADC). Pushing the ADC inside the processor chip enhances security, but this could be impractical. The confidentiality of input data from sensors is not important, because it is already accessible for everyone in the environment in the analog form. All the other data outside the scope of the main processor chip is encrypted, this include the external memory, transceiver, communication between WSN components on board, and the traffic in the wireless medium.

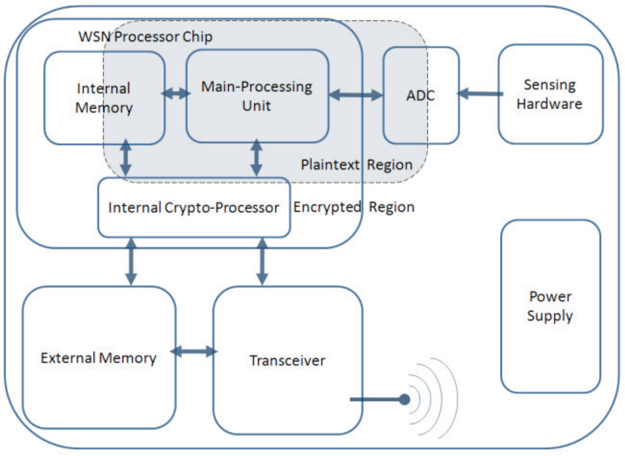

Fig. 2: Custom secure hardware platform for WSN

The overhead of decrypting data for aggregation and then encrypting it again has been addressed by researchers. Some of them suggested schemes based on homomorphic encryption, in which the encrypted data can be aggregated without the need of decryption, like the concealed data aggregation scheme in [7]. However, homomorphic encryption is malleable by design, and it's not suited for secure data transmission [8]. Our suggested scheme is more efficient in this matter, because it essentially minimizes the overhead of encryption by the use of efficient hardware implementations. This will make it more feasible to decrypt the data, aggregate it, and then encrypt it again. Sending all copies of encrypted data without aggregation is still a valid option. A comprehensive study of the packet size, the trade-off between the cost of decryption/encryption verses cost of transmitting data without aggregation has to be done to specify the threshold upon which we can decide if it is feasible to aggregate or not.

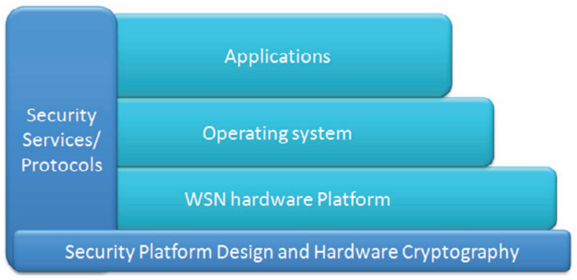
Applications
Security Services/ Protocols
Operating system
WSN hardware Platform
Security Platform Design and Hardware Cryptography

Fig. 3: Security View of Proposed Scheme.

The security view of our scheme is shown in Fig. 3. Our scheme can be seen as the first step to redesign hardware and software components of the WSNs from scratch with the security as main design goal. The additional design job and additional hardware for security platform will increase in manufacturing cost. We argue that a standardized robust design that is reliable enough to be used in a mass production manner is capable of reducing this cost.

4. Cryptographic Primitives review

The selection of cryptographic primitives for our proposed hardware platform should comply with WSNs constraints. The implementation of these primitives should be optimized for energy per encryption, static power consumption, and hardware area. Speed and throughput were the optimization goals for many literature implementations, but they are not considerable for WSNs because of their sparse and loose real-time communication. In this section, we will discuss the most efficient configurations and implementation methods of these primitives, their security status, and review their specialized implementations for WSNs in recent literature.

4.1. Symmetric primitives

Many symmetric algorithms exist in literature, the most efficient and secure ones are the AES finalists selected in second AES conference [9] to replace Data Encryption Standard (DES). The security and efficiency of AES finalists' algorithms has been intensively scrutinized by large number of cryptanalytic experts. In the third AES conference rijindael algorithm won the competition and was announced by National Institute of Standards and Technology (NIST) as AES algorithm.
Many hardware architectures for AES were proposed in the literature. Loop-unrolled architectures are the fastest, pipelined architectures are the best in terms of high throughput. However, the energy and area requirements of these architectures exceed the limitations of WSNs. The best choice for small area and energy constrained implementations are iterated architectures.
Two main variations between iterated architectures exist in the literature. The first one is the data-path width; 8-bit, 32-bit, and 128-bit architectures were proposed. Following the rule of thumb, higher data-path width architectures will have larger area and power requirements but smaller latencies. The authors in [10] provided energy per encryption comparison between the three data-path width choices, implemented on UMC 0.25 micrometer 1.8v hardware. As Table 2 shows, although 128-bit data path has the largest area and power usage; it has the smallest energy consumption per encryption. 32-bit architecture has medium area and power consumption, but it's the worst in terms of energy per encryption.

Table 2: Different data-path size comparison form [10].

Data-path width	Area (gates)	Power (mW)	Clock cycles	Throughput (Mbps)	Energy/Enc. (nJ)
8-bit	4023	1.06	160	8.1	169.6
32-bit	7732	3.73	54	23.7	200.5
128-bit	15980	11.61	11	116.4	127.7

Although 128-bit architecture has the least energy consumption per encryption; we consider the 8-bit architecture to be a better choice, because it's the best in terms of area and power. As a result, it will be the best choice in terms of static energy consumption (energy consumed while the hardware is idle).
The second variant between iterated architectures is the number of S-boxes used and their architecture. The 8-bit data-path design in Table 2 uses 2 S-boxes; and it completes one encryption in 160 cycles. The authors of [11] implement another 8-bit data-path architecture with one S-box. Their design has area size of 3400 gates, but it takes 1016 clock cycles to perform one encryption. Because of that the energy per encryption for his design will be higher than the design with 2 S-boxes in [10]. Larger number of S-boxes is used with higher data-path widths. However, this will lead to an increase the area and energy, and that is not efficient for constrained WSNs.

4.2. Hashing Functions

Most of WSNs security protocols use encryption algorithms with cipher block chaining CBC-MAC mode to produce the message authentication code. This method does not require implementing a separate MAC algorithm but it is not secure for variable-length messages [12]. Because of that we have chosen to use a standard hashing algorithm to produce the MAC. Another advantage is that the hashing algorithm can be run in parallel with the encryption algorithm which saves computation time and energy. The two most commonly used cryptographic hash functions in literature are MD5 and SHA-1. However, MD5 was recently broken with an attack against it used to break SSL in 2008 [13]. Some theoretical weaknesses in SHA-1 were discovered in [14] [15], and two successful attacks were reported in 2005. As a result, it was decided that it will not be used by USA government agencies after 2010. New security systems will be using more advanced hashing functions, such as SHA-2, or techniques that do not require collision resistance, such as randomized hashing [16]. NIST started a new competition to design a replacement for SHA-2 to ensure the long-term robustness of hash functions. The new algorithm will be given the name SHA-3, and it will become a FIPS standard in 2012.
Compared to 128-bit MD5 and 160-bit SHA-1, SHA-2 has larger message digests; i.e. 256, 384, and 512-bit. This will increase the energy consumed for transmitting the message digest, in addition to the increase due to additional hardware area of implementation. There are no specialized implementations SHA-2 for WSNs in literature at this time; most of its current implementations are optimized for speed and throughput.

4.3. Asymmetric primitives

Many security mechanisms proposed for WSNs in literature avoided asymmetric primitives, because of their high energy and time costs, especially for software implementations. The notion of infeasibility of these primitives has been changed partially due to the development of new asymmetric algorithms that are more efficient than RSA [17]. For example, the Rabin signature algorithm [18] is very similar to RSA; its main advantage is the speed of its encryption and signature verification operations. A disadvantaged is the signature size, a single signature requires 512 bits. NTRU [19] is much faster, for both encrypting and for verification operations than RSA. On the other hand, it also shares Rabin's scheme weakness; its signature requires 1169 bits.
The implementation in [5] is the first recognized hardware platform that is used to provide asymmetric encryption for WSNs. It was based on a commodity Trusted Platform Module (TPM) chip that extends the capability of a standard node. Although this implementation can perform 2048 bit RSA encryption in 500ms with 11mJ of energy; it has many shortcomings. Firstly, RSA has a large key size; this will result in large hardware area and energy consumption for the implementation. Secondly, transmitting a large key on the wireless medium is so expensive in terms. Thirdly, storing the keys on an EEPROM makes it easy to compromise the node by resolving the keys.
The Elliptic Curve Cryptography (ECC) [20] seems to be the most suitable asymmetric cryptographic primitive for WSN. The per-bit security for ECC is much more than other asymmetric algorithms. For example, the security of a 160-bit key for ECC is equivalent to 1024 bit key of RSA [21]. This relatively small key size result in smaller hardware area, shorter running time, and most importantly it saves energy needed to transmit the key in wireless medium. The energy cost of transmitting one bit is equivalent to the energy consumed in more than thousand CPU cycles of a standard WSN node processor [22].
ECC is based on algebraic concepts related with elliptic curves over Galois Fields. These fields can be binary fields $GF(2^n)$ or prime fields GF(P). For low energy hardware implementations binary fields are more attractive, since the operations engaged are only shifts and bitwise addition modulo 2. On the other hand, implementing arithmetic in prime fields is less efficient due to carry propagation. The use of redundant addition can overcome this issue [23].

Choosing the domain parameters of the elliptic curve is another matter of efficiency. The NIST and Securities Exchanges Guarantee Corporation (SEGC) proposed recommended curves and domain parameters in [21] [24]. Choosing a curve within a field size that is larger than 160 bits is recommended for both prime and binary fields. In July 2009, a 112-bit prime field ECC case was broken using a cluster of over 200 PlayStation 3 game consoles within 3.5 months [25]. Before that, a 109-bit key binary field case was broken in April 2004 using 2600 computers for 17 months [26]. The Certicom ECC challenge in [27] classified the curves in to two levels. Level 1 curves are considered feasible to break, and could be solved within a few months. Level II curves are considered computationally infeasible. Table 3 show the curve field sizes, their symmetric equivalent strength, and level classification.

Table 3: Security status of curves in GF(P) and GF(2^n).

Prime Fields	GF(112)	GF(128)	GF(160)	GF(192)	GF(224)	GF(256)
Strength	56	64	80	96	112	128
Level	I	I	II	II	II	II
Broken	In 2009	No	No	No	No	No
Binary Fields	GF(109)	GF(131)	GF(163)	GF(193)	GF(233)	GF(239)
Strength	47	64	80	96	112	128
Level	I	I	II	II	II	II
Broken	In 2009	No	No	No	No	No

The main ECC primitive operation is the scalar point multiplication. It consists of a series of point additions and doublings. Point addition and doubling consists of a series of modular addition, multiplication, squaring, and inversion operations according to a specific coordinate system. Many projective coordinate systems were proposed in the literature to avoid the high cost of modular inversion that is repeated in each point addition and doubling in the affine coordinate system. This is accomplished by projecting the ECC points to another coordinates that replaces modular inversion with modular multiplications. Many new curve representations associated with coordinate systems are being proposed. Montgomery coordinates is the most efficient so far [28]. Table 4 shows some common coordinate systems and their costs in terms of number of inversions I, multiplications M, and squaring S.

Table 4: Elliptic curve coordinates systems comparison.

Coordinate system	Doubling (I + M + S)	Addition (I + M + S)
Affine	1I + 2M + 2S	1I + 2M + 1S
Standard Projective	7M + 3S	12M + 2S
Jacobian	4M + 4S	12M + 4S
López-Dahab	2M + 4S	4M+2S
Montgomery	2M + 4S	4M + 1S

Redundant Interleaved modular multiplications [29], Barrett multiplication [30], and Montgomery modular multiplication [31] with Carry Save Addition are the most efficient modular multiplication algorithms proposed in the literature. Redundant Interleaved multipliers are reported to have some advantage over Montgomery's in area and speed. However, using one carry save adder with Montgomery algorithm makes it similar to Redundant Interleaved multipliers in area with small advantage over higher precision [32]. Barrett multiplication and Montgomery multiplication have similar area and timing results [33].
Some specialized ECC hardware implementations for WSNs have been proposed recently. In [34], an ECC processor is proposed in the 163-bit binary extension field. It uses Montgomery coordinators and Shamir's trick [35] to resist simple power analysis attack. The authors of [36] proposed a processor with special high speed inverter in affine coordinates, and with 4-bit digit-serial multiplier in 191 bit binary extension field. The processor has 25k gate count, 0.16 mm^2 chip area on 0.13µm CMOS technology and it can perform an ECC scalar multiplication in 342k cycles at 10MHz.

5. Proposed crypto-processor

The algorithms were coded in VHDL using the ModelSim digital simulation and functional verification tool. For the AES algorithm we have chosen the architecture proposed in [10] with an 8-bit data-path and two S-boxes. It takes 160 cycles to complete one encryption using this architecture. For the SHA-256 algorithm, the architecture was designed following the FIPS standard [37]; and it takes 80 cycles to produce the hash code of a 256 input block. The elliptic curve algorithm was implemented in the 163-bit binary extension field with redundant interleaved modular multiplier in the Montgomery coordinate system; 80 thousand cycles are needed to compute one elliptic-curve point multiplication. Another 8-kB dual port memory module was added to the design for secure key storage and data input/output. The crypto-processor design is shown in fig. 4.

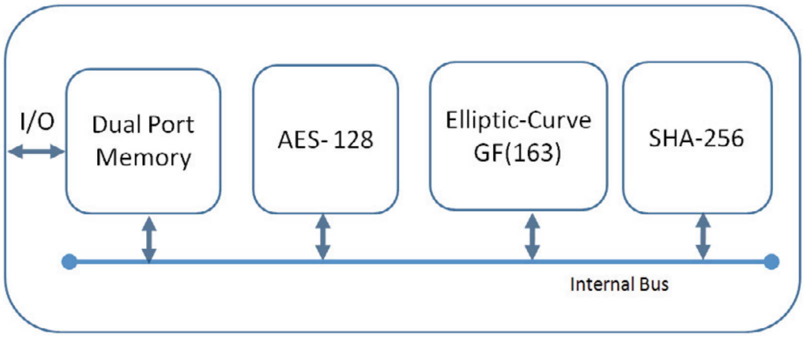

Fig. 4: crypto-processor design.

The energy and time results for the crypto-processor architecture using synthesizes for the Spartan 6 FPGA is shown in Table 5. The design consumed 4828 slices, which is 53% of the device resources. This leaves a space to add other components to implement a complete and secure sensor node processing unit as the one in fig 2.

Table 5: Implementation results

	Frequency (MHz)	Power (mw)	Clock Cycles	Time (µs)	Energy (nJ)
ECC-163			80000	967.3	16445
AES-128	92.67	17	160	1.727	34.64
SHA-256			80	0.863	38.18

Compared to related work our platform outperforms other designs in terms of the achieved level of security and security-related energy and time consumptions, which allows implementing highly secure wireless nodes without affecting the node life time of the wireless sensor network significantly.

6. Conclusion and Future Work

In this paper, we scrutinized the security holes in current WSNs platforms, and examined the current approaches to implementing the cryptographic primitives used to provide security services for these platforms. We also proposed a custom hardware platform for wireless sensor networks to fix these holes and provide more energy and time efficiency. We have chosen the most suitable cryptographic primitives for our platform based on their compatibility with constrained nature WSNs and their security status and reviewed their efficient configurations and specialized implementations for WSNs in the literature. In addition, we have implemented a crypto-processor that based on the most suitable architectures of these primitives provided energy area and time results.
In the future we will continue our work to implement a complete and secure wireless sensor node. This includes building robust security protocols and services based on the proposed hardware platform that minimizes the amount of energy consumption due to security related communication and processing.

References

1. P. Harrop, Wireless sensor networks 2009-2019. Technical Report, IDTechEx November, 2008.
2. E. Blaß and M. Zitterbart, Efficient Implementation of ECC for Wireless Sensor Networks, Telematics Technical Reports, University of Karlsruhe, March 2005. University of Karlsruhe, March 2005.
3. Joseph Bonneau, Ilya Mironov, Cache-Collision Timing Attacks Against AES. In proceedings of Cryptographic Hardware and Embedded Systems—CHES October, 2006.

4. M. Healy, T. Newe, and E. Lewis, Efficiently securing data on a wireless sensor network. Journal of Physics: Conference Series. vol. 76. 2007.
5. W. Hu, P. Corke, W. C. Shih, and L. Overs, secFleck: A Public Key Technology Platform for Wireless Sensor Networks. European Conference on Wireless Sensor Networks, pp 296-311, EWSN 2009.
6. J. Alex Halderman, et al., LestWe Remember: Cold Boot Attacks on Encryption Keys. Proc. 17th USENIX Security Symposium (Sec '08), San Jose, CA, July 2008..
7. J. Girao, D. Westhoff, and M. Schneider, CDA: Concealed Data Aggregation in Wireless Sensor Networks. Proceedings of ACM Workshop on Wireless Security (WiSe '04), Oct. 2004.
8. Akira Yamada, Wolfgang Schneider, Survey on the Current Status of Research and Development (R&D) of Cryptographic Technology in the European Commission. February, 2009.
9. James Nechvatal,Elaine Barker, Lawrence Bassham, William Burr,Morris Dworkin, James Foti, Edward Roback, Report on the Development of the Advanced Encryption Standard (AES), National Institute of Standards and Technology, October 2, 2000.
10. Lian Huai, Xuecheng Zou, Zhenglin Liu, Yu Han, An Energy-Efficient AES-CCM Implementation for IEEE802.15.4 Wireless Sensor Networks, International Conference on Networks Security, Wireless Communications and Trusted Computing. April, 2009.
11. M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, AES Implementation on a Grain of Sand IEE Proc, vol. 152, pp.13-20, Oct. 2005..
12. M. Bellare, J. Kilian, and P. Rogaway., " The security of Cipher Block Chaining. Advances in Cryptology Crypto '94 Proceedings, Lecture Notes in Computer Science Vol. 839, Springer-Verlag, Y. Desmedt, Ed., pp. 340-358, 1994.," in .
13. Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, MD5 considered harmful today: Creating a rogue CA certificate. December 30, 2008.
14. X. Wang, Y.L. Yin, H. Yu, Finding collisions in the full SHA1. Crypto 2005.
15. M. Sugita, M. Kawazoe and H. Imai, Grobner, Basis Based Cryptanalysis of SHA-1, Cryptology ePrint Archive, Report 2006/098, 2006.
16. Shai Halevi and Hugo Krawczyk, Strengthening Digital Signatures via Randomized Hashing. Advances in Cryptology - CRYPTO '06. LNCS vol. 4117, pages 41-59. Springer, 2006.
17. Rivest, R., A. Shamir, L. Adleman , A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21 (2): 120–126. 1978.
18. M. O. Rabin, Digital Signatures and Public Key Functions as Intractable as Factoring Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.
19. Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, NTRU: A Ring Based Public Key Cryptosystem. In Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, J.P. Buhler (ed.), Lecture Notes in Computer Science 1423, Springer-Verlag, Berlin, 1998.
20. Blake, G. Seroussi, N. P. Smart, Elliptic Curves in Cryptography. Cambridge University Press, ISBN 0-521-65374-6, 2000.
21. National Institute of Standards and Technology. Recommended Elliptic Curves for Federal Government Use. August, 1999.
22. Holger Karl, Andreas Willig, Protocols and Architectures for Wireless Sensor Networks.John Wiley & Sons: Chichester ISBN: 978-0-470-09510-2.pp-44 June 2005.
23. E. Savas, A.F. Tenca, C¸ .K. Koc, A scalable and unified multiplier architecture for finite field GF(p) and GF(2m), Proceedings of the Cryptographic Hardware and Embedded Systems—CHES, LNCS, vol. 1965, Springer, Berlin, 2000.
24. Standards for Efficient Cryptography Group (SECG). Recommended Elliptic Curve Domain Parameters. SEC 2, september, 2000.
25. http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml Fact Sheet NSA Suite B Cryptography, U.S. National Security Agency.
26. D.J. Bernstein, Irrelevant patents on elliptic-curve cryptography.
27. Certicom ECC Challenge. November 10, 2009 update. http://www.certicom.com/images/pdfs/challenge-2009.pdf.
28. D. Bernstein and T. Lange, Faster addition and doubling on elliptic curves, Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science 4833, Springer-Verlag, New York, 2007.
29. G. Blakley, "A Computer Algorithm for Calculating the Product A.B modulo M," IEEE Transactions on Computers vol. C-32, no. 5, pp. 497–500, May 1983.
30. P. D. Barrett, Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. Advances inCryptology - Crypto'86, LNCS, vol. 263, pp. 311–323. Springer-Verlag, 1987.
31. P. Montgomery, Multiplication without trial division. Mathematics of Computation, vol. 44, pp. 519–521, 1985.
32. D. N. Amanor, C. Paar, J. Pelzl, V. Bunimov, and M. Schimmler, Efficient hardware architectures for modular multiplication on FPGAs. in Proceedings of FPL 2005.
33. M. Knezevic, L. Batina, and I. Verbauwhede, "Modular Reduction without Precomputational Phase," In Proc. IEEE International Symposium on Circuits and Systems (ISCAS). May, 2009.
34. Lejla Batina, Jorge Guajardo, Tim Kerins, Nele Mentens, Pim Tuyls, Ingrid Verbauwhede, Public-Key Cryptography for RFID-Tags. PerCom Workshops, 2007.
35. J. Solinas, Low-weight binary representation for pairs of integers Centre for Applied Cryptographic Research, University of Waterloo, Combinatorics and Optimization Reseach Report CORR 2001-41, 2001.
36. H. Aigner, H. Bock, M. H¨utter, and J. Wolkerstorfer., A low-cost ECC coprocessor for smartcards. In Cryptographic Hardware and Embedded Systems - CHES 2004, LNCS 3156, pp.107-118, 2004.
37. NIST, FIBS-PUB 180-2, Secure Hash Standard, August, 2002.