International Journal of Research in Advanced Engineering and Technology
Online ISSN: 2455-0876; Impact Factor: RJIF 5.44
www.engineersjournals.com
Volume 1; Issue 3; December 2015; Page No. 49-53

A study on wireless sensor networks and the concepts of encryption and key exchange

[1] Gangu Gayatri, [2] Dr. Vaibhav Bansal

[1] Research scholar, Department of computer science, OPJS University, Churu Rajasthan, India
[2] Assistance professor, Department of computer science, OPJS University, Churu Rajasthan, India

Abstract
Encryption and key distribution are important primitives to build secure Wireless Sensor Networks (WSN). Different block ciphers were proposed in literature to provide encryption in resource constraint distributed networks. A large amount of different key distribution schemes were implemented, targeting different types of WSNs. These schemes face issues with respect to their requirements, implementations, and theoretic foundations. In this paper we provide an overview of selected encryption schemes and introduce so called fault attacks targeting hardware implementations of block ciphers.
Selected key pre-distribution schemes, like random key pre-distribution and hierarchical key distribution, are presented and depicted. Memory efficient Elliptic Curve Cryptography (ECC) introduces feasible public key cryptography schemes for WSNs. A summary of selected ECC based schemes is given with a focus on Identity Based Encryption (IBE), which is especially useful in scenarios with a trusted sink node.

Keywords: Cryptography, key exchange, encryption

Introduction
Wireless Sensor Networks (WSN) are increasingly deployed in environments were data integrity, confidentiality and authentication of senders becomes an important requirement. Considering a WSN forming a Body Area Network (BAN), several low-cost nodes are attached to the human body to collect sensor data. In most body networks, this data is periodically transferred to a sink node for further processing. Sink nodes are also often designed to work as gateways to transfer data to e-Health systems residing in the cloud. These BANs could be deployed in hospitals or at homes for ambient and assisted living monitoring elderly.
Patient data is often classified as high privacy information that must be transferred in an encrypted form, without revealing sensitive contents. Security implementations in BANs must also ensure that nobody can inject faked data as this can have serious impacts, e.g., wrong prescription of drugs. Thus data packets need to be authenticated to allow a verification of their origin. While an attacker could have state-of-the-art hardware to perform attacks, sensor nodes are constraint embedded platforms, which results in an unusual challenge regarding encryption and key exchange.
To keep energy consumption low, nodes have limited computing power, small RAM, and low storage capacity. Thus, classic public key cryptography based on RSA trapdoor function is not suitable due to its high computational overhead. The complexity of RSA operations does not scale linear with their key size [1], which makes them too costly when initialized with parameters defined as sufficiently secure by the National Institute of Standards and Technology (NIST) [2] or other standard institutions.
As today's key management schemes heavily rely on public key cryptography, researchers have proposed several lightweight alternatives to RSA based key management. Once keys are exchanged and authenticated, efficient block ciphers are required to encrypt network communication in real time. Besides block ciphers, which were designed for a small memory footprint and smaller block size, modern microcontrollers used in sensor nodes come with implementations of the Advanced Encryption Standard (AES) [3].
In our scenario, where a WSN is attached to a body and collects patient data, special issues need to be resolved. These WSNs face interoperability issues and disruptive connections between its nodes, meaning that an end-to-end connection is not always available. Especially in the case of data transmission between meeting humans, a connection is only available ad hoc at random meetings or at specific meeting times.
Therefore approaches exists to deploy network stacks implementing a Delay-Tolerant Network (DTN) [4]. We provide an overview of software implementations for encryption. We study software modules for Tiny OS2 [5] and Contiki3 [6] and look at their choice of block ciphers and operation modes for authenticated encryption.
Furthermore, precautions regarding hardware implementations are presented and an introduction into fault attacks is given. Besides presenting the challenges of using block-ciphers, the following sections mainly focus on key management as it often introduces new and interesting problems. An overview of key pre-distribution schemes is given in this paper, where random key pre-distribution and hierarchical group key management is presented in detail. We look at schemes using public key cryptography based on Elliptic Curve Cryptography (ECC).

Encryption
Because of RAM and performance constraints, block ciphers need to be optimized regarding memory consumption and execution time when implemented for sensor nodes. Also energy consumption and protocol overhead have to be kept small for better adaption into these distributed networks. We argue that the selection of appropriate block ciphers should not compromise on security, as attackers conceivably are

equipped with powerful hardware. It is also imaginable that attackers capture encrypted traffic for later analyzation, which contradicts the argument that sensor networks are short lived and thus would require only small key sizes. Hence, we have a critical look at statements regarding security tradeoffs.

Available Software Implementations

In the following, available software modules for encryption are presented and evaluated regarding their choices of block ciphers and authentication. Tiny Sec [7] for Tiny OS 1.x was one of the first implementations of link layer security in wireless sensor networks. It ensures low power requirements, small latency and low bandwidth overhead and supports two different modes of operation: authenticated encryption (Tiny Sec-AE) and authentication only (Tiny Sec-Auth) [7].

CBC mode with an 8 byte Initialization Vector (IV) is used together with a block cipher. Stream ciphers were excluded as they are vulnerable to repeating IVs and an addition of more bytes to a packet would violate their premise of keeping the packet sizes small. To overcome the problem of data leakage in CBC mode using repeated IVs, they propose to pre-encrypt the IV before use. As an appropriate energy-efficient block cipher, they chose Skipjack due to its simple implementation and efficiency.

To ensure message integrity CBC-MAC is deployed. Even from yesterday's perspective, Skipjack was a bad choice as it is a subsequently declassified algorithm, designed behind closed doors at the NSA, vulnerable to several cryptanalysis attacks [8]. "In a footnote, which was added later to their paper [7], they themselves had to admit that AES would also be a viable choice with similar performance like Skipjack. Even Tiny Sec successor Mini Sec [9] and another implementation called Tiny Key [10] are based on Skipjack instead of widely accepted and standardized block ciphers, like AES [11]."

Implementations for Contiki

Contiki Sec [12] provides security for the operating system Contiki with the following modes of operation: Contiki Sec-Enc providing encryption only using AES with a 2 byte IV, Contiki Sec Enc uses Cipher Block Chaining-Ciphertext Stealing (CBC-CS) mode. This mode is utilized to prevent expansion of the produced ciphertext, when messages are not entirely separable into input blocks. This method requires a single network wide key, pre-distributed on all devices or any method providing a session key resulting from other protocols. Contiki Sec-Auth Authentication only is provided by producing a MAC using the standardized CMAC algorithm. Contiki Sec-AE AE mode provides both authentication and encryption by deploying AES in Offset Codebook Mode (OCB), which generates an authenticated cipher stream. All choices are well made as the authors chose well-studied algorithms mostly standardized by the NIST. No practical attacks against AES itself is known, also CBCCS mode for encryption only and CMAC for authentication only are sufficiently secure regarding their requirements. OCB mode for combined authentication and encryption is also the best choice for this purpose, because formal analysis has shown its security and superior performance compared with for example AESGCM [13]. What should be noted is that OCB mode is patented and its authors allows its use only free of charge when implemented in software licensed under the GNU GPL, non-commercial software, or government software 4.

Hardware Implementations

Since the NIST standardized the Advanced Encryption Algorithm (AES), manufactures have implemented it in hardware, providing extended instruction sets for software modules [3]. Many improvements were made to use less hardware for AES specific operations. The radio controller Atmel AT86RF231, for instance, provides instructions to setup AES using Electronic Codebook (ECB) mode or Cipher Block Chaining (CBC) mode.

Ghaznavi *et al*., for example, redesigned the Mix Columns function to use less hardware. Others optimize regarding parallelizing the algorithm in hardware [15]. Consequently, it is not necessary to implement the block cipher algorithm in software. However, this does not mean that no mistakes can be made. The provided implementations have to be used cautiously to prevent for example key exposure, injection/fault attacks, or side-channel attacks. In the following, we want to focus on selected challenges and attacks regarding these hardware implementations.

About the Utilization of Hardware Instructions

When implementing encryption using microcontroller instruction sets, e.g., instructions provided by Atmel AT86RF231 [3], some precautions have to be taken. ECB mode must not be used because of its insecurity against replay attacks and preservation of message pattern due to its non-chaining structure. As AES in CBC mode itself does not provide any authentication, additional software implementations are needed to provide tamper resistant ciphertexts. CCM mode (counter with CBC-MAC) or AES-GCM can provide authenticated encryption, standardized in RFC 5084 [16].

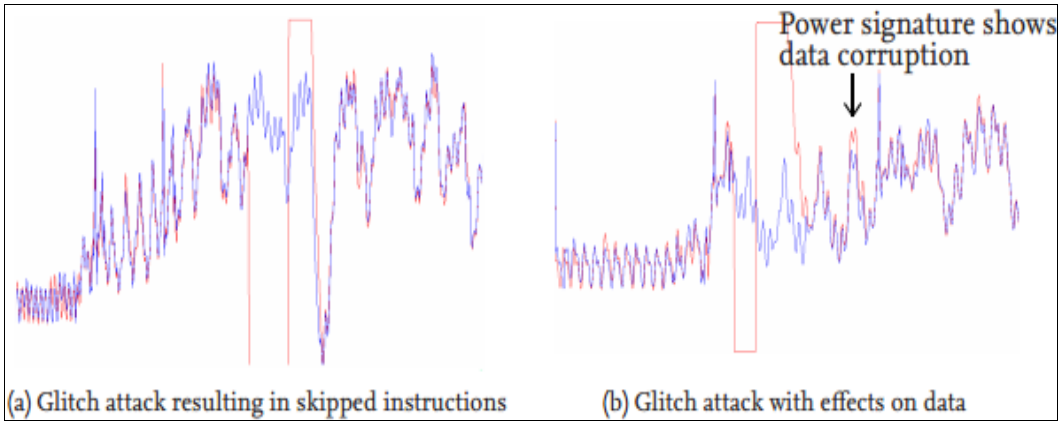
Fault Attacks against AES Hardware Implementations

Considering an attacker who has physical access, fault attacks become possible. Fault attacks influence hardware components by changing their environment to cause faults in their calculations. Common methods of inducing faulty behavior by external stimulation according to the sorcerer's apprentice guide to fault attacks [17] are: Variation of power supply & Changing the voltage that is supplied to a microcontroller can cause glitches, like misinterpretation or skipping of instructions.

This attack is only possible with direct physical access. Temperature Hardware must operate in a specific temperature range, as manufactures do not guarantee correct viability outside these thresholds. Because read and write temperature thresholds on non-volatile memories vary, attacks can be mounted to heat chips until read operations fail, but write operations work according to the specification. The photoelectric effect causes microcontrollers to be sensitive regarding white light. Laser With similar effects as white light, lasers can be targeted much more precisely to influence specific areas of a chip. Hagai Bar-El and Whelan have done experimental evaluations to show that instructions are skipped, when the power supply of their microcontroller was dropped from Vcc to 0 V and later resumed [17]. As shown in Figure 1a, a selective execution of instructions can be achieved. When executed at exact moments, this attack can cause serious security implications for the implemented encryption scheme. The induced instruction glitches can reduce the number of loops in a block-cipher to produce an easy to break single-

round variant. Besides reducing loops, data can be manipulated when the voltage supply is controlled precisely (cf. Figure 1b). Because many smart-cards are vulnerable to these types of attack, much research is being done to find new attacks and countermeasures.

Roche *et al*. [18] attack AES implementations based on a new fault model. Fault attacks combined with physically leaked side-channel information, e.g., electromagnetic radiations, are used to recover the key. The attack will induce faults in the last round of AES key-scheduling.



Source: [17]

Fig 1: Experimental fault attacks by dropping voltage supply, blue curve depicts normal execution, red curve depicts execution with V_{cc} glitch

Countermeasures

Countermeasures against fault attacks can take place on hardware and/or software level. Sensors can be implemented to actively detect malicious induced effects and classify them as attacks. This includes light sensor, frequency detectors, and supply voltage detectors. Hardware redundancy, based on duplicated circuits and comparison of results, can make attacks much harder. These are one of the most implemented and effective countermeasures when done the right way [7, 8].

All hardware countermeasures could also be implemented in software, but would double the computing time in most cases, as parallelization is not as easy as in hardware implementations [7].

Key Pre-distribution

To allow the use of previously discussed block ciphers, encryption keys need to be pre-distributed among the devices that want to communicate with each other. Algorithms for deploying these pre-shared keys need to be implemented, which must support different requirements based on the setup of its WSN. According to Xiao *et al*., key distribution schemes should mainly provide authenticity to provide a way to identify nodes, scalability, and flexibility to allow adding of new nodes at every time.

Authenticity, scalability, and flexibility lead to several basic attack scenarios key distribution schemes have to protect against. An attacker could, for example, compromise certain parts of the network and replicate those nodes to take over the entire network. Key distribution schemes should resist against those attacks by preventing replication. When bad behavior of single nodes is detected, those nodes should be excluded, by revoking them using the key management scheme.

Furthermore, it should be possible to deploy new nodes into existing key management infrastructures and to retain secrecy of benign nodes, when single nodes are taken over (Resilience). Several key pre-distribution schemes were proposed to fulfill these requirements, which are based on differing assumptions regarding the sensor network. We will present fundamental problems, the most common schemes, and show possible weaknesses.

The simplest key pre-distribution form is the pre-deployment of a single network wide key on all nodes in the network. Besides small storage requirements, this scheme apparently does not prevent any of the three discussed attacks as a single compromised node results in a compromised network. In a pair-wise key distribution scheme, a node is deployed with n − 1 keys to communicate with every other node. This offers pair wise encryption and authentication as every key belongs uniquely to two nodes.

Pair wise pre-distribution also complies with Resilience, because a takeover of nodes does not reveal secret information about other nodes due to differing keys. This scheme only makes sense in small sensor networks as each new node requires a new key to be stored by all other nodes, resulting in a high memory usage. A superior sink node with a larger memory and higher transmission rates can be used as Key Distribution Center (KDC). A KDC can be queried for session keys used one time for a specific communication between nodes.

Random Key Pre-distribution

Eschenauer and Gligor presented a random key pre-distribution protocol to overcome the problem of storing N − 1 keys on every participant. The distribution scheme works as follows:

1. A large pool of |S| key-identifier pairs is generated.
2. K key-identifier pairs are randomly taken out of this pool, where K N and N is the number of nodes in the WSN and saved as a keyring. Every node gets its own keyring assigned.
3. Based on the identifiers, two nodes can agree upon one shared key, while the probability is sufficiently high that they both share at least one matching key.

If no shared key is available the nodes perform path-key discovery. This means a third party node will be searched that shares keys of both communicating nodes. The scheme

successfully optimizes regarding storage space and is also scalable, as the size of the underlying key pool and the number of chosen keys can be adaptively chosen according to the size of the network. The downsides of this schemes are that it provides no form of multicasting messages because between the sender and every recipient a different key has to be negotiated. There is also no method that would provide strong revocation of compromised nodes and freshness of keys.

Schemes Based on Random Key Pre-distribution

Several more complex protocols were proposed that are based on the random pre-distribution model. Du *et al*. published a model extending the concept using Blom's key matrix instead of individual keys. A matrix Mn×n is generated, where nodes can select keys from by looking at specific cells. If node i and j want to communicate, i looks at Mij and j looks at Mji to select the same shared key. In this scheme n matrices are generated and on each node only randomly selected matrices are stored, like in the basic random key pre-distribution scheme. Their scheme offers more robustness and security, because more nodes need to be compromised compared to the basic random pre-distribution scheme. This advantages comes to the cost of more computational overhead and memory consumption.

Even more complex protocols like LEAP and SHELL were published. They introduce different key types, like individual, group, cluster, and pair wise keys, where each key type is distributed and used differently. They fulfill most requirements and are flexible, but require complex implementations. Consequently much memory space is needed for their implementation and keys. Besides their memory usage, their implementation will likely contain bugs due to their high complexity.

Hierarchical Group Key Management Schemes

Panja *et al*. were the first who proposed a hierarchical group key management. A hierarchical management scheme consists of groups sharing the same group key, ordered hierarchically based on their computing power or other factors like access rights. The scheme consists of cluster heads, having many general sensor nodes as children, arranged in a tree structure.

A new group key is calculated, when a cluster head broadcasts an encrypted authenticated message containing the instruction to refresh the group key. This could also contain the instruction to remove a compromised node, which implements revocation. In addition to these algorithms the authors differentiate between intra-cluster group keys and inter-cluster group keys. When the cluster head broadcasts the key the first time, blind factors are added, which unique numbers are assigned to individual nodes.

When distributing the group key, each node can recognize its own blind factor and replace it with their own real factor. This prevents attackers from eavesdropping the group key in deployment phase. This scheme is flexible, because it is possible to refresh group keys and revoke nodes. However, revocation only works to a certain degree, because cluster heads can only revoke leafs or complete branches. The downside is that it is assumes that a cluster head does not get compromised, which makes it a worthy target. It is also quite elegant and simpler to implement than over-engineered schemes like Shell [9].

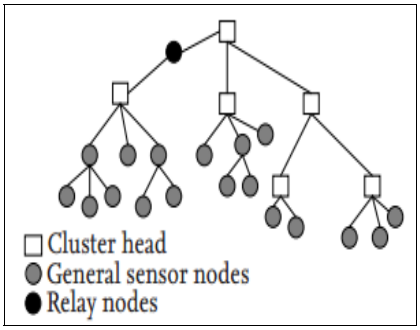


Fig 2: Hierarchical structure proposed by Panja *et al*. (Derived from [22])

ECC Based Key Management

Key pre-distribution schemes do not require large amount of processing power or RAM, but are much less flexible than schemes based on public key cryptography. In the past, most sensor networks were deployed with pre-distributed keys, lacking important requirements like fast revocation of keys and secure integration of newly deployed nodes. RSA based public key algorithms were omitted due to their seemingly large power consumption while performing underlying multiplication methods. New implementations of Elliptic Curve Cryptography (ECC) were done to provide usable public key cryptography with low requirements. ECC is based upon the Elliptic Curve Discrete Logarithm Problem (ECDLP), which states that it is hard to find a discrete logarithm of an elliptic curve element with a publicly known base point.

Gura *et al*. have done implementations of public key cryptography for RSA and ECC for sensor nodes based on ATmega128 and CC1010 microcontroller [1]. Their implementations are highly optimized regarding memory access, as this is the most limiting factor on these small processors. The key size for ECC keys can be much lower than RSA based keys, e.g., a 1024 bit RSA key pair provides as much security as a 160 bit ECC key pair [2]. This is due to the fact that RSA trapdoor function is based on the hardness of integer factorization and ECC is based on its ECDLP.

While sub-exponential algorithms exits to solve integer factorization, known algorithms to solve ECDLP scale exponentially [1]. Their ECC point-multiplication using 160 bit, implementing SECGs standardized elliptic curves in conjunction with their optimized multiplication, is by a factor of two faster than RSA-1024. As an example by absolute numbers, RSA-1024 private key operation needed 10.99 s while a 512 bit montgomery exponentiation for ECC needed 5.37 s. ECC-240 outperforms RSA-2048 by a much more higher degree as shown in their paper [1]. Their evaluation has shown that it is indeed possible to use public key algorithms without hardware acceleration on microcontrollers intended for sensor nodes. As recommended key sizes by NIST [2] for 2011-2030 are RSA-2048 and ECC-240, ECC should be favored due to their small key sizes and low runtime of its algorithms.

Szczechowiak *et al*. improved the work of Gura *et al*. Besides implementing a hybrid multiplication algorithm, they chose Koblitz curves as these are less expensive due to their appearance without point doubling. Their implementation is

done for Tiny OS and written in C/nesC with some optimizations using assembly, generated by utility programs for better portability.

Conclusion

In this paper, we presented selected security primitives for wireless sensor networks. Software implementations of encryption modules and fault attacks against hardware implementations were discussed. Key pre-distribution schemes were depicted with their advantages and disadvantages. As a promising public key technique, ECC could solve some problems pre-distribution schemes could not solve perfectly and is thus viewed as WSNs future key distribution implementation. Several hard issues remain unsolved.

The problem of adding nodes to an existing network is difficult, even with public key cryptography, as every node in this distributed network has to be informed about newly added and accepted public keys. This can partially be solved by defining the sink node as a trusted third party, as done in Identity Based Cryptography.

Revocation is even harder, because of several pitfalls. Firstly, a node needs to be identified as a malicious node, possibly overtaken by an adversary. Only a few papers deal with this issue, as it is impossible to detect malicious nodes, when they behave correctly and just capture traffic. Secondly, the revocation packet needs to be routed to every node quickly, although malicious nodes could work as black holes, preventing the distribution. These problems exist due to the distributed nature of wireless sensor networks. Many WSN are deployed in areas without any additional surveillance, which makes fault attacks possible and leave them undetected.

References

1. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, Sheueling Chang Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In: Lecture Notes in Computer Science 2004; 3156:119-132.
2. Elaine Barker, William Barker, William Burr, William Polk, Miles Smid. Recommendation for key management. In: NIST Special Publication 800-57 Part 1 Rev. 2011, 3.
3. AT86RF231/ZU/ZF datasheet. Atmel Corporation.
4. Fall K. A delay-tolerant network architecture for challenged internets. In: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications. ACM, 2003, 27-34.
5. Levis P, Madden S, Polastre J, Szewczyk R, Whitehouse K, Woo A *et al*. Tiny OS: An operating system for sensor networks. In: Ambient intelligence, 2005, 35.
6. Dunkels A, Gronvall B, Voigt T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In: 29th Annual IEEE International Conference on Local Computer Networks. IEEE, 2004, 455-462.
7. Chris Karlof, Naveen Sastry, David Wagner. Tiny Sec: a link layer security architecture for wireless sensor networks. In: Proceedings of the 2nd international conference on Embedded networked sensor systems, SenSys '04. Baltimore, MD, USA: ACM, 2004, 162-175.
8. Biham E, Biryukov A, Dunkelmann O, Richardson E, Shamir A. Initial observations on the skipjack encryption algorithm. In: SAC, 1998, 98.
9. Luk M, Mezzour G, Perrig A, Gligor V. Mini Sec: a secure sensor network communication architecture. In: 6th International Symposium on Information Processing in Sensor Networks, IPSN. IEEE, 2007, 479-488.
10. Doriguzzi Corin R, Russello G, Salvadori E. Tiny Key: A light-weight architecture for Wireless Sensor Networks securing real-world applications. In: Eighth International Conference on Wireless on-Demand Network Systems and Services, WONS, 2011, 68-75.
11. Felix Büsching, Andreas Figur, Dominik Schürmann, Lars Wolf. Utilizing Hardware AES Encryption for WSNs. In: Proccedings of the 10th European Conference on Wireless Sensor Networks, EWSN, 2013, 33-36.