On Security Issues in Embedded Systems: Challenges and Solutions

Lyes Khelladi, Yacine Challal, Abdelmadjid Bouabdallah, Nadjib Badache

On Security Issues in Embedded Systems: Challenges and Solutions

Lyes Khelladi, Yacine Challal, Abdelmadjid Bouabdallah, and Nadjib Badache,

***Abstract*—Ensuring security in embedded systems translates into several design challenges, imposed by the unique features of these systems. These features make the integration of conventional security mechanisms impractical, and require a better understanding of the whole security problem. This paper provides a unified view on security in embedded systems, by introducing first the implied design and architectural challenges. It then surveys and discusses the currently proposed security solutions that address these challenges, drawing from both current practices and emerging research, and identifies some open research problems that represent the most interesting areas of contribution.**

***Index Terms*—Security, Embedded Systems.**

I. Introduction

RECENT advances in digital electronics and wireless communications have enabled a widespread proliferation of embedded systems which are becoming more ubiquitous within our daily lives. Ranging from micro-sensors, smart cards, cell phones and PDAs to network routers and flight control computers, embedded systems are continuously revolutionizing critical sectors like industry, finance, science and home life; leading to an increased number of potential applications [78, 79].

During their operation, embedded systems are often required to store, access, or communicate data of a sensitive nature, making security a serious concern. Indeed, the success and adoption of several next-generation applications and services are predicted on the ability of embedded systems' designers and manufacturers to ensure adequate security, and gain the trust and confidence of consumers, suppliers and involved government institutions. For example, 2.5G and 3G wireless applications including mobile commerce (m-commerce) [79] that involves several types of embedded systems like cell phones, or PDAs require a high level of security. In fact, security is cited as the single largest concern among surveys of prospective m-commerce users [80].

Thanks to Internet, security has long been a concern in such computing and communication systems; and substantial research efforts have been devoted to address it. While the knowledge and experiences gained from prior works including cryptographic algorithms and security standards give us a head start in the quest to secure embedded systems, there are several features inherent to these emergent computer systems that still need to be addressed: first, The need to make the embedded devices mobile or encapsulated in a larger system require them to fulfill their functions with limited hardware resources, mainly characterized by a scarce memory (static and volatile) and reduced CPU processing capability. These limitations pose tight constraints on both communication and computing capacity. Second, some types of embedded systems being a microelectronic devices, can only be equipped with a limited batteries (< 0.5 Ah, 1.2 V). In some application scenarios, replenishment of power resources might even be impossible [81]. In such case, adopted mechanisms directly affect the system lifetime, and must take the energy efficiency as a first design objective. Third, many embedded systems operate stand-alone in a non-controlled environment. As a result, they generally tend to face extreme operating circumstances like vibration, shocks, lightings, power supply fluctuations, user abuse and so on. This close physical coupling with the operating environment necessitates the use of reliable mechanisms and introduces additional security consideration that will be emphasized in the next section. And finally, the design process of an embedded system is highly influenced by the production cost. In large volume application (e.g., cell phones, automobiles) saving just a few cents per unit can add up - literally to millions of dollars. In more cost-sensitive application like sensor networks, decisions increasing the cost of individual sensors directly influence the overall technology's feasibility [81]. All the unique features cited above impose new challenges for security design in embedded systems; these characteristics should gain a more significant attention among research community, so that ripe and effective security solutions can be achieved.

In this article, we provide a unified and recent view on the challenges facing security in embedded systems. We also survey security solutions that have been proposed thus far for this particular class of computer systems. Our aim is to provide a better understanding of the current security issues through a comprehensive study that covers almost all functional aspects of embedded systems including se-

CERIST
Algiers
Phone: +213 21 91
Fax: +213 21 91
email: lkhelladi@mail.cerist.dz
Compiegne University of Technology
Royallieu Research Center - Heudiasyc lab.
Postal Code: 60200, Compiegne, France
Phone: +33 3 44 23 44 23
Fax: +33 3 44 23 44 77
email: yacine.challal@utc.fr
Compiegne University of Technology
Heudiasyc lab. France
Phone: +33 3 44 23 52 50
Fax: +33 3 44 23 44 77
email: madjid.bouabdallah@utc.fr
USTHB
Algiers
LSI lab.
Phone: +213 21
Fax: +213 21
email: badache@wissal.dz

curity for local computations, end-to-end communications and embedded networking. We also attempt an investigation into pertaining design and architectural aspects and outline the use of certain methods to meet secure design objectives.

The remainder of this article is organized as follows: section 2 introduces the reader to the baseline security objectives in embedded systems. In section 3 we outline the design challenges that rise from various embedded systems security requirements. In section 4 we survey and discuss emergent solutions that address these challenges and finally section 5 summarizes the insights gathered in this work and concludes with future avenues of research.

II. Background: Security goals in embedded systems

Targeted security services in embedded systems are not altogether different from those of other computer systems. Their goal is to protect sensitive data and/or resources from different attacks and misbehaviour threats. The four main security objectives in embedded systems include:

- Availability: ensures that the desired system's services are available whenever they are expected, in spite of the presence of attacks. Availability mechanisms in embedded systems seek to combat denial of service and energy starvation attacks, as well as other tampering attacks that will be explained further.
- Confidentiality: guarantees that the secrecy of transmitted data between communicating parties is maintained. i.e., no one other than the legitimate parties should know the content of the messages being exchanged.
- Authentication: represents the process of verifying an identity claimed by/for a system entity. The objective of this security service is to prevent a malicious party from masquerading as someone else. And,
- Data Integrity: that protects data against unauthorized changes, including both intentional alteration or destruction and accidental change or loss, by ensuring that such changes to data are detectable.

III. Security challenges in embedded systems

The sensitive application fields of embedded system often require them to provide critical functions that could be sabotaged by malicious entities. However supporting security services emphasized in the previous section translates to various design challenges that rise from the unique features of embedded systems and their specific application requirements. These challenges make it impractical to directly use the conventional proposed solution, and move security considerations from an afterthought into mainstream design issue [82, 83].

In this section, we describe the various challenges involved in supporting security on embedded systems; our objective is to provide a full integrated view of all factors that are driving the design of secure embedded systems. These factors are important because they serve as a guideline to design security architectures and protocols, and they can be used to compare different proposed schemes.

The relationship between embedded systems features, the challenges they induce and the corresponding proposed solutions is illustrated in figure 1. This latter provides to the reader a general overview on various concerns about security in embedded systems, it serves also as a roadmap that we will use in this paper to explore the discussed issues.

A. Processing performance

The limited processing and memory capacity of embedded systems make it impossible for their architectures to keep up with the continuously growing complexity of security mechanisms and the increasing data rates offered by recent communication networks. This problem is much more noticed in systems that need to process very high data rates such as network routers, or in low-end systems equipped with scarce processing and memory resources like PDAs, cell phones or smart cards (table I.

Figure 2 redrawn from [84] plots the processing requirements in MIPS (million of instructions per second) of symmetric encryption algorithms 3DES and AES, along with integrity algorithms SHA and MD5, for various data rates characterizing different network technologies. The "composite" curve illustrate the workload needed by a security protocol, like SSL [85], that involve combined usage of symmetric encryption and message authentication algorithm during bulk data exchange. We can easily conclude from the given figure that low-end embedded system's processors such as StrongARM SA-1110, having processing capability around 150 mips, can only sustain a data rate lower than 50 mbps for a composite security protocol; assuming that this processor can be fully dedicated for the security operations. This means that any higher data rate is unattainable by such processors, leading to the so-called "processing performance gap" [82]. Indeed, the challenge is to minimize the processing performance gap by the development of more efficient security architectures that adequately combine between hardware and software through co-design approaches [86–88]. Moreover, a judicious choice of the most suited basic security primitives and cryptographic algorithms may help to enable better security processing performance for resource-constrained embedded system.

B. Power consumption optimization

Power expenditure constitutes one of the most challenging design factors that Bottleneck the development of security schemes in battery-powered embedded systems. This factor needs to be considered independently from other hardware limitation concerns, since it can directly affect the overall system's life-time such in mica2dot sensors [89] where replenishment of batteries is not possible.

However, conventional security mechanisms tend to be conservative in their security guarantees, typically adding a large number of message overhead and computation which induces high energy consumption. In fact, Potlapally et al. [90] show that the introduction of encryption in net-

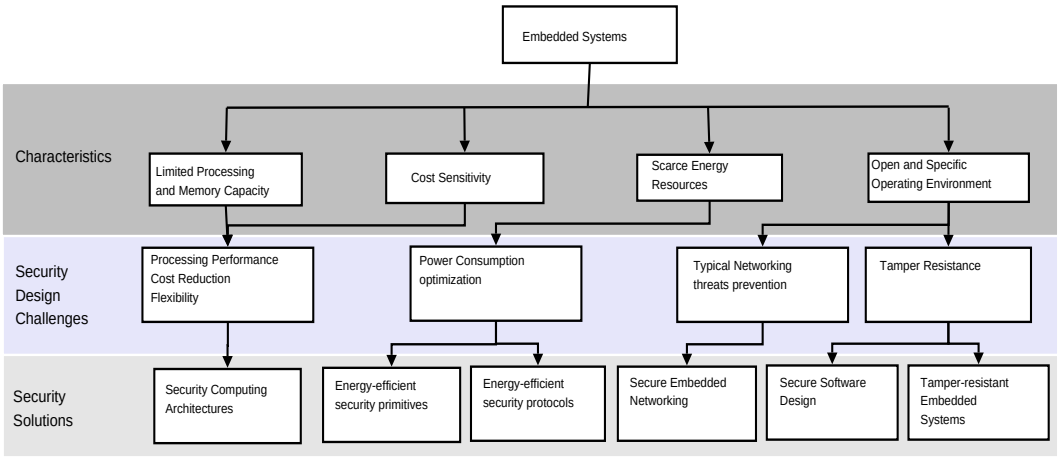

Fig. 1. Security roadmap for embedded systems

	Smart dust Sensors [52]	Typical Smart Cards	iPAQ PDA [113]
CPU	8 bits, 4MHz	8 bits, 3.57MHz	32 bits, 206MHz
Flash memory	8KB	16KB	64MB
RAM	512B	1KB	16MB
Bandwidth	10Kbps	9.6Kbps	11Mbps

TABLE I
HARDWARE LIMITATIONS IN EMBEDDED SYSTEMS

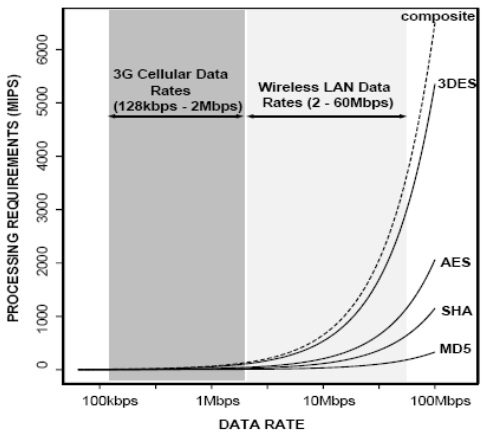

Fig. 2. Processing requirements of cryptographic algorithms at different data rates

worked sensors' communication may decrease their battery life by a factor of 80%. This is mainly due to several elements: first, the use of cryptographic primitives implies additional complex computations for the embedded system's processors during the key setup and data encryption/decryption phases. Moreover, the energy consumption will also be influenced by the increased size of the transmitted bulk data after their encryption. And finally, the use of secure communication necessarily implies broking up the data transfer into several sessions, where each one requires additional transmissions for authentication and key establishment stages; this fact will significantly help in very fast draining of the battery capacity.

To face this challenge, energy-efficient security protocol execution is highly required; this objective can be achieved in multiple ways. The obvious one consists of making the execution of employed cryptographic primitives more efficient through a combination of a new hardware and software optimization techniques. Although this solution is naturally a suitable way to minimize energy dissipation of security mechanisms, it is not always evident to realize, and usually involve an overhead in the form of an increase in silicon area or more complex software. The second alternative is to adapt existing security solutions and make them energy-efficient by allowing them to alter their tasks and to choose the best combination of their constituent building blocks depending on the operating environment. This behavior adaptation must be guided by certain rules that realize the best trade-off between the guaranteed security level and the available energy resources.

C. Tamper resistance

The theoretical strength of the utilized cryptographic primitives does not reflect in any case the security level of the embedded system. This later also depends on the quality of the primitives' implementation, and by that, their tamper resistance. In addition to the classical cryptanalysis approach where a cryptographic primitive is viewed

as an abstract mathematical object, using a secret value called "key", nowadays attackers mostly investigate the alternative view given by the implementation of the primitive. Here, the primitive manifests itself as hardware circuit or as a program that will run on a given embedded processor, and will thus present very specific characteristics [91]. Such a view implies that security protocols and cryptographic algorithms can be broken by exploiting detected implementation flaws, or simply by observing properties related to their implementation like timing behavior, or energy consumption. These new attack trends have been essentially spurred by the close physical coupling of embedded systems with their environment which makes them more susceptible to tampering. Moreover, the use the weakly secure wireless communication medium by a high majority of embedded devices and the new communication paradigms used in embedded networks have introduced additional attacks that will be discussed further (Fig. 1).

Basically, tamper resistance issues in embedded systems deal with the following two classes of threats:

1. *Physical and Side-channel attacks:* This class integrates all attacks that interact with the embedded system's hardware and exploit the physical-side of system implementation flaws and proprieties. Two subcategories can be distinguished [92]: invasive attacks like micro-probing techniques that generally require getting access to the chip level components in order to interfere and manipulate with system internals. Such attacks use a relatively expensive infrastructure and are, by consequent, hard to deploy. On the other hand, non-invasive attacks don't require the device to be opened and can be achieved in several forms. For example: fault induction attacks observe the targeted system's behavior after generating errors or failures on it, through manipulation of its operating conditions like (supply voltage, temperature, radiation, light, etc..). Other attacks possibility includes side-channel attacks [93] like power analysis, or timing attacks that rely on observing the correlation that exist between the device power consumption, or its timing profile, and the executed operations or the manipulated secret data (cryptographic keys). With the same manner, electromagnetic analysis attacks try to intercept and measure the electromagnetic radiation emitted by a device to reveal some sensitive information about the executed cryptographic primitives or the utilized secret keys.
2. *Software attacks* These major threatening attacks are leveraged by the system's capability to download, upgrade, and execute application codes. They are essentially based on malicious software tools (like viruses and worms) and exploit the logical-side shortcomings of the embedded system implementation. Here also, and similarly to physical attacks, we can enumerate several types [94]:
   - *Interception-based attacks* try to passively eavesdropping sensitive data in order to compromise the users' privacy or the confidentiality of exchanged data. An example of such attacks could be the use of a logical analyzer for inner line bus probing of an embedded system.
   - *Interruption-based attacks* target the system's availability by making it unusable. This could be done through an energy exhaustion attack that exploits the scarce energy resource in battery-powered systems and make them undergo additional workload in order to drain out their batteries. Such denial of service attacks are also privileged in low-end embedded systems with limited processing and storage capacity where the device can rapidly fail in the case of intensive workload.
   - *Modification-based attacks* compromise software integrity by exploiting detected vulnerabilities. A common example is the use of the buffer overflow vulnerability to overwrite the stack memory, and thereby, transfer control to a malicious program whose execution can have undesirable effects.

D. Typical networking threats prevention

The development of communication and networking technologies leveraged the idea of wireless embedded networks that may fully or partially rely on different kinds of embedded devices, providing them a broader range of more ubiquitous applications. However, and from the security point of view, the wireless networking of embedded systems has essentially introduced two additional challenges. First, the broadcast nature of the wireless radio signal makes the embedded network extremely vulnerable to several networking threats. Indeed, any malicious party can easily carry out his attack without requiring any physical access to the networks' deployment perimeter. One example of such threats is the "parking lot" attack [95]. Strong countermeasures are therefore needed in the embedded device to overcome the insecure nature of the communication medium, and data link layer security becomes more critical than in the wired world. The second security challenge is to prevent many typical threats which are closely related to the embedded networks' applications and the communication paradigm that they employ. For instance, the infrastructureless property of ad hoc networks and the participation of end-user embedded devices into the network services introduce a large number of routing security issues that must be taken into account by every wireless communication-enabled embedded system.

E. Design flexibility

There exist several important factors that make flexibility a fundamental requirement for secure schemes design in embedded systems. In fact, the security architecture is often required to contain a large variety of security protocols and standards so that the embedded system can (i) guarantee the multitude of security objectives (users' authentication, data confidentiality, digital right management, etc.), (ii) facilitate interoperability in different environment (eg. a PDA that needs to work in both 3G cellular and wireless

LAN networks). Furthermore, and since hacking methods and skills are in continuous evolution, the security solution (iii) should allow upgrading enclosed mechanisms or adding new ones if necessary.

F. Cost reduction

As emphasized earlier, embedded systems are often highly cost-sensitive, even few cents can make a big difference when building millions of unit. However, integrating top-level security is not always cost-effective for embedded systems because it mandates the use of more expensive hardware and software, and requires a long time rigorous design process along with environmental failure protection and testing [96]. Consequently, the designer's responsibility consists of balancing the security requirement of an embedded system against the cost of implementing the corresponding security measures.

IV. Security solutions for embedded systems

After that we have described the embedded systems' features along with the security issues and challenges which they induced, we present, in this section, the different security solutions that have been proposed thus far in the literature. In Section A we first discuss the need to consider security requirements in early stages of the software design process in order to prevent implementation flaws and enhance the system's tamper resistance. Other techniques related to physical and logical tamper-resistance are also highlighted in section B. Afterward, the section C analyzes possible architectural enhancement that have been developed to improve security processing performance. Further, section D tackles the important energy consumption challenge in battery-powered embedded devices by exploring the possible methods that allow energy-efficiency while securing embedded systems. And finally, section E emphasizes some threats that are typical to specific embedded networking applications, and exposes related mitigation solutions.

A. Secure software design

Although modern cryptography has enabled the embedded software to provide a relatively robust defense against "conventional" attacks that target basic security requirement such as confidentiality or integrity, more efforts are still needed at higher levels to protect the embedded software from a large diversity of attacks which exploit their development defects essentially caused by implementation bugs or design flaws. Considering such threats is as important as the need to integrate hard-to-break mechanisms that meet functional security objectives, because the embedded system's strength depends on the easiest way to attack it, and this latter is mostly done through a discovered design or implementation shortcoming.

Consequently, preventing such threats requires a good software engineering practices and involve thinking about security concerns early in the software development life cycle. While today's engineering methods are well suited for developing complex functional systems, they are often poorly suited to the task of preventing undesired functionality. As a result, latent security problems have always been kept undetected with conventional testing methods. Becoming aware of the stated problem, practitioners are increasingly convinced that waving security considerations as deeply as possible into the Software Development Life Cycle (SDLC) constitute the most suitable solution.

In [97] the national Institute of standards and technology NIST provides a general framework for incorporating security consideration throughout SDLC, including a minimum set of security concerns that need to be considered within each of the phases composing the presented development life cycle.

As shown in figure 3, there exist various secure design practices that apply at different levels of the development cycle:

1. *The design level*: in this phase, a risk assessment is mandatory to identify possible threats and vulnerabilities in the system. Based on their security impact, and the cost of the related mitigation techniques, designers establish an acceptable and cost-effective level of risk for the software. This operation requires "threat modeling" which is the core step in any security solution, and can be achieved using various formal and semi-formal specification techniques like data flow diagrams or UML diagrams.
2. *The implementation level*: secure coding is the most important requirement during this level. For that, developers need to be adept in the basic tenets of coding secure applications in order to mitigate related vulnerabilities like buffer overflow and format string vulnerabilities. Further, the use of secure and certified development tools such as secure compilers described in [99, 100] is desirable for better security.
3. *The testing level*: before the final penetration tests performed by specialized teams, a formal security evaluation stage is needed. This latter is accomplished through static source code analysis tools or dynamic runtime testing methods [100, 101], for example, the fault injection systems can also be used to check for the presence of flaws.

B. Tamper-resistance techniques in embedded systems

In addition to the secure design practices explored previously, mitigating implementation-related threats in embedded systems passes through the integration of appropriate tamper resistance techniques that strengthen the device against software and hardware attacks. These techniques serve as reinforcement to baseline security services ensured by cryptographic primitives and protocols; they are used for either attack prevention, detection or recovery [102]. Moreover, in some cases, it may be desirable to preserve an irrefutable persistent record of the attack on the embedded systems; this requires the existence of tamper-evident mechanisms that can not be reversed by malicious entities. In this section, we survey various tamper resistance techniques by classifying them into two categories, depending on the considered tampering attack nature which can be

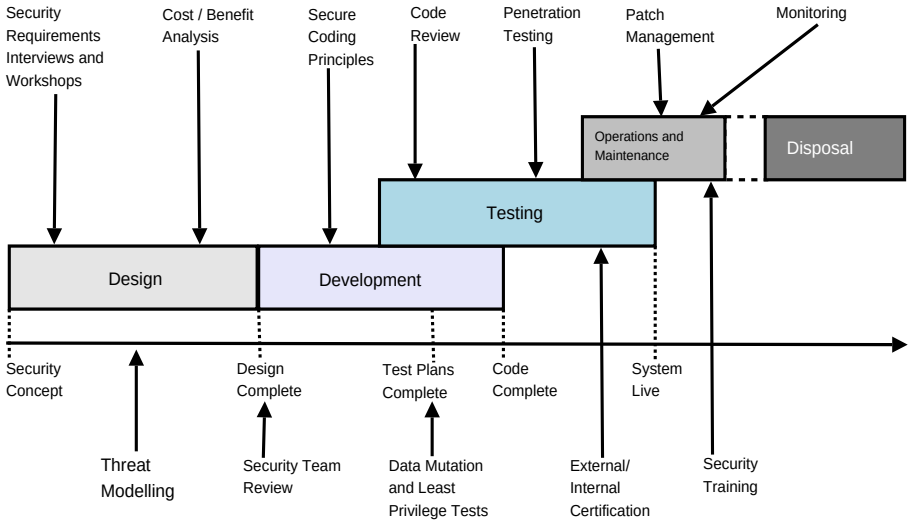

Fig. 3. Secure Software Development Life Cycle [98]

logical or physical.

B.1 Mechanisms against physical and side-channel attacks

The Federal information processing standard FIPS140-2 [96] provides four increasing qualitative levels of physical (as well as other) security requirements intended to reflect different degrees of tamper resistance. Security level 1 represents the minimum physical protection with production-grade enclosure that may be removable; level 2 requires the use of tamper-evident covers or seals which allow the detection of a physical compromise, while level 3 mandates the use of more advanced tampering detection and response mechanisms. For example, it requires the use of zeroization circuitry that immediately zeroize all plain text secrets and private keys located in memory when valid or invalid access is detected through the system's doors, removable covers or maintenance interface. Finally level 4 adds to the requirement of previous levels the condition that the system must include environmental failure protection (EFP) or undergo environmental failure testing (EFT). Concerning side-channel attacks, several prevention software and hardware approaches have been proposed to mitigate symptoms that allow the leak of the system's side-channel information like power dissipation, timing and electromagnetic radiations [91, 93]. Software-based countermeasures include randomization of the instruction execution sequence, introducing dummy instructions, balancing Hamming weights of the internal data and bit splitting to name a few [103–105]. On the hardware level, randomization can also be applied on the clock signal or the power consumption [1, 2]; moreover, aggressive shielding techniques as well as methods that break the locality of chip layout (permit the spreading of the chips' components across their entire surface) are effective in defeating electromagnetic analysis attacks [3]. Although it has been shown that software-based countermeasures are versatile and the most efficient, they considerably hinder performance of cryptographic algorithm in term of energy, memory and execution time. Hence the challenge to achieve is the development of side-channel mitigation techniques with as little extra cost as possible. Besides, we can find in the literature several solutions that have been proposed for particular physical attacks prevention. For instance, the use of concurrent error detection methods for transient fault attacks [4] or the adoption of environmental sensors for fault injection attacks and recovery triggering [5].

B.2 Mechanisms against software attacks

Countermeasures against software attacks typically target one or more of the following objectives [93]:

- Ensure privacy and integrity of sensitive code and data during every stage of software execution in an embedded system.
- Determine with certainty that it is safe from a security standpoint to execute a given program.
- Remove security loopholes in software that make the system vulnerable to such attacks.

Similarly to physical attacks, defeating software attacks can be done through software and hardware-assisted mechanisms. The latter class consists essentially in the implementation of secure co-processors dedicated to processing all sensitive information in the system; in addition to the design and maintenance of selected area of memory (volatile or non-volatile, off-chip or on-chip) as a secure storage location whose the access should be restricted to trusted programs only [6–8].

Furthermore, software-based solutions cope with a broader range of areas that we summarize in the following domains:

1. *Secure bootstrapping:* Providing an acceptable guarantee about the security of the operating system's starting is paramount in every effective security solution. Without such a secure bootstrap neither the executed application nor the underlying operating sys-

tem kernel can be trusted, since they are invoked by an untrusted process. A common approach adopted by the proposed techniques in this area is to use the notion of trust boundaries across the system's hardware and software components; this notion allows the detection of illegal accesses carried out by malicious software. Early works that have exploited this approach is described in [9], where a hierarchical solution is provided by exploiting the layered nature of the boot process. Starting from power on state, the system can move to the next layer, and thus expand the trust boundary, only if the integrity checks of the current layer success.

2. *Software authentication and validation:* Similarly to the operating system, every known application in the embedded system should undergo a validation step before its execution. This can be done through computing a hash of the application code and comparing it to a pre-computed golden value. The validation process can also be extended to the application's runtime behavior by using oblivious hashing technique [10] which calculates the checksum of the application's execution traces.
   Concerning the untrusted codes, that correspond to newly installed applications for example, the most efficient way might be to execute them in a restricted environment similar to sandboxes provided by several virtual machines like Java Virtual Machine JVM. Other possible techniques include requiring the code supplier to bundle safety proofs with the program executable [11], or detecting malicious codes by monitoring all control transfers in the program in order to verify for any security policy violation.
3. *Operating system (OS) and application enhancement:* Many countermeasures can be achieved in the system level to protect it from software-tampering attacks. For instance, the trusted Computing Group (TCG) and Next Generation Secure Computing Base (NGSCB) initiatives [13, 14] relies on different OS modifications which protect sensitive code or data. Amongst them we cite strong process isolations so that the private resources of one process can be protected from another process; process-level attestation that authenticate the code before that it establishes communication channel with corresponding device; the use of cryptographic file system to ensure secure storage; in addition to several modifications to context switching procedures, exception handling, memory management, etc. it is important to note that many of these changes would require architecture-level modifications for security.
   In the application level, adopted techniques include the use of encryption wrappers that allow dynamic run-time encryption/decryption of the software code to prevent static code analysis attacks; code obfuscation [12] that consists of transforming the application's code so it becomes less intelligible to humans, and thus preventing reverse engineering; in addition to Digital Right Management (DRM) and software watermarking [13] techniques that embed certain controls in the software to protect it from illegal widespread utilization .

C. Security computing architectures

Security solutions in the architectural level consider the mapping of adopted algorithms and protocols within a layer of software and hardware specializations. One of the main objectives of these solutions is to address the processing performance gap emphasized in section B by enhancing the processing capabilities of the embedded system through adequate combination of software and hardware. In fact, security processing in early proposed schemes was performed by executing security software on general-purpose processors embedded in the system. While these software-based approaches provide a good flexibility by allowing multiple implementations and upgrading of security mechanisms, they are not efficient in term of processing performance [15, 84].

C.1 Cryptographic Hardware Accelerators

One approach that overcomes the processing inefficiency of the software-based solutions is to completely implement the resource-greedy cryptographic computations on a dedicated hardware using ASICs (Application Specific Integrated Circuits). With this manner, the embedded processor can offload the cryptographic computations, through high speed bus for example, to the custom hardware designed to guarantee higher processing speed and lower energy consumption. Various companies offer Cryptographic Hardware Accelerators that implement diverse asymmetric and symmetric ciphers for systems ranging from low-power mobile appliance and smartcards to high-performance network routers [6, 16, 17]. Although this "hardwired-algorithm" approach has proved its excellent performance level, it's much less effective in term of cost and flexibility, when several cryptosystems are required in order to support multiple security protocols and emerging standards for better interoperability with other systems.

In an attempt to achieve a trade-off between processing performance, flexibility and design cost, a third generation solutions have been proposed, they can be classified into two subcategories: processors enhancement with (i) special purpose extensions, and (ii) general purpose extensions.

C.2 Embedded processors with special purpose extensions

This approach denoted by Application Specific Instruction Processors (ASIP) consists of tightly integrating the accelerator's hardware circuitry in the processor core it self, and invoke it with special instructions. A typical example is to implement one round of a symmetric cipher such as DES (Digital encryption Standard) within the processor hardware and invoke it using a custom instruction. This solution allows one to guarantee enhanced performance for the intended specific cryptographic algorithm but misses flexibility. Indeed, no processing accelerations could be achieved for any other cryptographic algorithm.

An illustrative example of ASIP processors is presented in [18] where custom instructions are used to perform a selected set of symmetric, asymmetric and hushing primitives.

C.3 Embedded processors with general-purpose extensions

In order to provide better flexibility than in the previous ASIP-based approach, designers chose to improve the processing capability of general-purpose processors with instructions set that are indeed general-purpose primitives. These instructions usually execute versatile operations that potentially compose more complicated functions employed in different cryptographic algorithms or other resource-greedy applications. For example, the studies which focused on symmetric bloc ciphers design identified arbitrary bit permutations as a useful operation that runs very slowly in existing processors [19,20]. As a result, various bit permutation instructions have been proposed, as in [21] where any one of the n! possible permutations of an n-bits word can be achieved through O(1) instructions rather than O(n) cycles in the case of conventional processor instructions' use.

On the other hand, asymmetric cryptography has totally different general-purpose instruction primitives, these later tend to enhance modular exponentiation used in several encryption algorithms like RSA and Diffie-Helleman by accelerating a high number of modular integer multiplications on large operands eg. 1024 bit numbers. Newer cryptographic schemes such as Elliptic Curve Cryptography (ECC) [22] are however based on computationally faster operations that consist of polynomial binary multiplications rather than integer multiplications. Moreover, they need smaller operands (keys) to achieve equivalent level of security. Consequently, the trend in the implementation side has been to implement dual field multipliers that can efficiently perform integer and binary multiplications on the same circuit [23].

Pax [24] is a typical example of new Reduced Instruction Set processors that integrate general-purpose extensions to accelerate both symmetric and asymmetric cryptographic computation. Such an approach is quite promising for embedded processors in smartcards and networked sensors where flexibility is required with small form factors and low energy consumption.

C.4 Security Protocol Engines

Security processing is not limited on cryptographic computations only; it involves also security protocols operations, like packet header/trailer parsing, classification etc. these operations often consume a significant portion of the system's processing capability and their execution need to be optimized. While network processors have been designed to accelerate the processing of traditional network protocols, "security protocol engines" have been proposed to accelerate security protocol computations. Moreover if programmable, theses engines can be used to execute multiple protocols efficiently allowing better flexibility. For example, the 7811 security processor from HIFN [25] can be used in VPNs to perform IPSEC processing at high data rate (250mbps). Again, it can provide high performance support to several layer 2 protocols' execution such as PPP [26] and PPTP [27].

D. Energy-efficient security schemes

Addressing power limitation issues in embedded systems can be done either by enhancing the hardware capability of the embedded system (improving supply), or by reducing the energy consumption of security mechanisms (alleviating demands). On the supply side, and beyond the improvement of battery's capacity whose growth doesn't exceed 8% per year [82], architectural solutions described in the previous section, are of great importance. They considerably reduce the energy dissipated by security processing through the optimisation achieved in the execution workload and the use of energy-efficient hardware operations [28]. At the other side (demand side), adapting the existing security mechanisms to the energy-constrained environments in embedded systems, or proposing new energy-efficient approaches is paramount.

In this section, we present energy efficient security mechanisms in embedded systems regarding to two functional levels: the first considers the primitives' level or the protocol's building blocks: minimizing energy expenditure in this level mainly consists in carefully choosing the employed algorithms (cryptosystems, hash functions, random number generators, etc.) and adequately adjusting their tuning parameters and operation modes. The second level deals with security protocols themselves which represent a coarse-grained control flow exchanged between the communicating parties to establish a certain degree of mutual trust. Energy efficiency in this level is mainly affected by the combination methods between the cryptographic primitives, and the overhead induced by exchanged control messages.

D.1 Energy-efficient cryptographic primitives

Several studies and benchmarks have been conducted to evaluate and compare the energy consumption induced by various cryptographic primitives (symmetric/asymmetric ciphers, hash functions) in a variety of embedded architectures [29–33, 90]. Having the objective to identify the most suitable techniques that should be adopted to ensure the embedded world security, researchers have deeply investigated different aspects of these primitives and studied their impact on the system's energy expenditure. Beside the fact that different algorithms and hardware architectures have been considered in each work, almost all of them concluded that some trade-off should be achieved between the desired security level and the available energy resources. In this section, the operational parameters of security primitives that allow establishment of such trade-off are discussed. Moreover, an insight is provided to some symmetric and asymmetric encryption algorithms, widely agreed as energy-efficient.

1. *Symmetric encryption*

Symmetric encryption algorithms can be either stream or bloc ciphers. In the former class the plaintext is encrypted one bit at time. This class is appropriate for applications where buffering is limited (e.g., cell phones), or when characters must be individually processed as received. Since most stream ciphers used in practice tend to be proprietary and confidential, relatively few fully-specified algorithms exist in the open literature; one of the well known ciphers is RC4. By contrast to the first class, block ciphers tend to simultaneously encrypt fixed-size blocs of the plaintext message using a fixed encryption transformation. Examples of bloc ciphers include 3DES [108], AES [109] and blowfish [77].

For the Encryption/decryption operations, both stream and bloc ciphers use one input secret key which is expanded, during a key setup phase, in order to derive a distinct and cryptographically strong key for each round. A round denotes the repeated sequence of mathematical computations that form the overall cipher's algorithm.

Concerning the energy efficiency of symmetric ciphers, it's difficult, or even impossible to characterize one algorithm as more energy efficient than other. However, based on undertaken works [33,34,90], we can extract the following pertinent observations, which are illustrated in table II:

- Although stream ciphers like RC4 are known to be faster and more efficient compared to block ciphers, they present a more significant encryption/decryption energy cost. This is mainly due to the energy consumed in memory accesses resulting from buffering misses.
- Some secret key encryption algorithms can exhibit a great contrast in dissipated energy between the key setup and the encryption/decryption phases. For example, Blowfish is one of the most optimal symmetric ciphers in term of energy consumed during encryption/decryption operations, however, it has a very high energy cost for the key setup phase which encompasses complex operations involving 512 iterations and 4168 bytes of generated sub-keys. Such cryptographic algorithms could be the most appropriate for applications that don't require frequent session establishment, and leverage high volume of exchanged data.
- If we consider the overall cipher's functioning, that include both key setup and plaintext encryption/decryption phases, AES appears as a judicious choice because of its competitive energy costs. Moreover, AES cryptanalytic properties have been well studied [35–37], allowing it to combine energy efficiency with acceptable security level.

2. *Asymmetric ciphers*

Unlike symmetric ciphers, asymmetric cryptosystems use a pair of different, though related, public and private key to encrypt and decrypt plaintext. In these algorithms, guessing the secret key from the public one is equivalent to solve a computationally hard mathematical problem. For example, in RSA, the hardness of this operation is based on that of integer factorization, while in DSA, and Diffie-Hellman (DH) [111], it is based on that of discrete logarithm problem in integer fields. The encryption/decryption algorithms in asymmetric cryptosystems are also based on a more complex, and harder to implement mathematical operations than in symmetric cryptography. For example, the basic operation in RSA, DSA and DH is modular exponentiation [38]. The complexity of such operations has largely limited the utilization of this class of cryptosystems in resource constrained embedded systems, since they generally require important processing capability and high energy consumption (table III). Consequently, most of the studies [34, 39] report that symmetric key ciphers and hash functions are between two to four order of magnitude faster than digital signatures and public key cryptography. It has been concluded then, that these techniques become the systematic tools of choice for low-end resource-constrained embedded systems.

Nevertheless, some works tried to revoke the thesis stated about the unfeasibility of public-key cryptography in low-end embedded systems. In fact, Gaubatz et al. [31] show, in the context of sensor networks, that some low-complexity asymmetric cryptosystems like Rabin's scheme or ECC, can be implemented in a low-power co-processor that handles all of the compute-intensive tasks. Authors have also pointed out that such low-power design solutions tremendously simplify the implementation of many typical security services using public-key cryptography, and additionally reduce the transmission power due to less protocol overhead induced by some operations in symmetric ciphers like shared key agreement. We recall here that such hardware-assisted solutions miss flexibility. That is, even if intuition may support their utilization in homogeneous sensor networks, their application remains questionable in other embedded devices which perform in more heterogeneous environments, where several asymmetric protocols should be enabled.

Further, a software-only approach is proposed in [40] to improve computational efficiency, and hence the energy consumption of public-key encryption algorithms. The main idea consists of using some known algorithmic optimization like Chinese Remainder Theorem [41, 43] and Montgomery Multiplication [42], along with new advanced techniques such as input block size selection in order to allow better performance of modular exponentiation-based asymmetric ciphers like RSA, El Gamal and DH. The formulation of various optimization methods led authors to propose a formal "algorithm design space" that is defined by various possible algorithm configurations. It has been demonstrated that the algorithm's performance can vary significantly (over an order-of-magnitude) across this space, depending on input data characteris-

	DES	3DES	IDEA	CAST	AES	RC2	RC4	RC5	BLOWFISH
Key Setup μJ	27.53	87.04	7.98	37.63	7.87	32.94	95.97	66.54	3166.3
Enc/Dec $\mu J/byte$	2.08	6.04	1.47	1.47	1.21	1.73	3.93	0.79	0.31

TABLE II

ENERGY CONSUMPTION OF SYMMETRIC CIPHERS ON IPAQ 3870 PDA [90]

Processor	Power (mW)	Frequency (MHz)	Energy consumption (mJ)			
			RSA		DSA	
			Enc	Dec	Sign	Verif
MIPS R4000	230	80	16.7	0.81	9.9	20.0
SA-1110 "StrongARM"	240	133	15.0	0.74	9.1	18.2
MC68328 "GragonBall"	52	16	840	42	520	1040
MMC2001 "M-Core"	81	33	137	6.9	85	169
ARC 3	2	40	1.13	0.06	0.70	1.40

TABLE III

ENERGY CONSUMPTION OF ASYMMETRIC CIPHERS FOR DIFFERENT EMBEDDED PROCESSORS [32]

tics and underlying hardware architectures. Although this work focused on processing performance that are fully proportional to the devices' energy consumption, similar studies are required to explicitly show the impact of such optimizations on the dissipated power. Moreover, we think that the use of such algorithmic optimizations in emergent asymmetric cryptosystems should be investigated. In fact, some newer asymmetric ciphers are continuously gaining attention among the research community because of their low-complexity operations, and their potential for offering acceptable security level at reduced key sizes. We believe that the use of these cryptosystems in low-end embedded systems is more promising. Hereafter, we briefly describe a prominent example of them, namely ECC algorithm.

Elliptic Curve Cryptography (ECC): ECC [22] operates on a group of points on an elliptic curve defined over a finite field. This curve is characterized by its equation, and a fixed parameter G called base point. The core of elliptic curve arithmetic operations is scalar point multiplication which computes Q=kP (a point P on elliptic curve is multiplied by an integer k, resulting in another point Q on the curve). This computation is achieved through a combination of point additions and point doublings. For example 7P can be expressed by 7P = 2((2P) +P) +P. The security of ECC relies on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP) which states that Given P and Q, it is computationally hard to find k. Then, the large random integer k acts in ECC as a private key, while the result of its multiplication with the base point G serves as the corresponding public key.

Besides the fact that ECC is based on lower-complexity operations compared to standard public-key methods, it can offer equivalent security with substantially smaller key sizes. e.g., a 160-bit ECC key provides the same level of security as a 1024-bit RSA key. As a result, NIST recently approved ECC for use by the U.S government [110]. Moreover, several standards organizations, such as IEEE, ANSI, OMA(Open Mobile Appliance) and the IETF have ongoing efforts to include ECC as recommended security mechanism. Finally, we note that several cryptographic protocols have further been proposed using the elliptic curve cryptography such as Elliptic Curve Diffie Hellman (ECDH) key exchange, and Elliptic Curve Digital Signature Algorithm (ECDSA) which are the counterparts of DH and DSA algorithms respectively.

3. *Energy consumption vs. security level* Certain cryptographic algorithms, particularly symmetric ciphers, can provide different security levels, each with its associated energy consumption characteristics. In such case, the provided security level can be adapted with the current state of the battery on the system, in order to achieve the best trade-off between system's life time and the guaranteed security level. It has been shown through experimental studies [90] that security level and power consumption on handheld devices like a PDA can be affected by several functional parameters that we describe below:
   - *Operation mode:* Bloc cipher algorithms can have different modes of operation which result in variants with different power dissipation characteristics and security levels. The simplest mode is Electronic Codebook Mode (ECB), other modes are Cipher Bloc Chaining (CBC), Cipher Feedback mode (CFB) and Output Feedback mode (OFB), all of which employ a different feedback mechanism which makes the encryption of current plaintext bloc dependant on the result of previously encrypted blocs. With the feedback mechanism, even for the same key, identical plaintext blocs will not always have the same ciphertext [38]. This makes CBC, CFB and OFB more resistant to cryptanalysis attacks than ECB. On the other hand, ECB consumes much less energy than other modes. This conducts designers to use ECB in applications where it can offer comparable security as other modes, like situations where encrypted plaintexts are relatively short, and contain low number of identical blocs.
   - *Key size* The key size is one of the most critical factors that determine the security level of a given cryptographic algorithm. In fact, the larger the key size, the greater the security offered. How-

ever, it has been shown in [44] that for the same algorithm, the power consumption increases quasi-linearly with the key length. As expected, larger keys cause more workload during all the algorithm phases, thus leading to higher energy consumption. Further, since the key length parameter affects more the key setup phase than the encryption/decryption one, algorithms spending less energy for the key setup stage will be less sensitive to this factor in term of energy consumption than others.

- *Number of rounds* As mentioned earlier, a typical encryption algorithm operates in several rounds that represent repeated execution sequences with identical structures and different sub-keys. Therefore, increasing the number of rounds implies the use of more different sub-keys. This has the same effect on the primitive's security level as the key length increase. As a result, using lower number of rounds costs less energy, but exponentially decreases the offered security level. This confirms the fact that as far as the trade-off between security and power consumption is concerned, high security and low power can't be achieved at the same time.

D.2 Energy-efficient security protocols

By a security protocol, we mean every protocol that allows several embedded systems to communicate in a secure manner, with guarantee of base-line security goals such as sensitive data confidentiality or entity authentication. Depending on the system's nature (Cell phones, PDAs, sensors, etc.) and their corresponding applications (GSM, WLANs, etc) several security protocols have been developed. Although that some of these protocols are tuneable, and can be adaptively executed to optimize their energy dissipation, rare are those that considered energy efficiency as an early design objective.

On the other hand, several works have been conducted to explore the possibility of adapting conventional security protocols, initially proposed for wired networks, by altering their behavior for embedded devices, so that an acceptable trade-off can be achieved between the security level and corresponding energy requirements.

For example, Potlapally et al. [90] presented a comprehensive analysis of the energy consumption of the transport layer security protocol SSL [85] on iPAQ PDAs. They studied the impact of various functional parameters like adopted cipher suites, key lengths and data transaction size on both processing and communication stages of the protocol. Moreover, the effects of some optional operation modes, like mutual authentication enabling, on the overall energy consumption of SSL have been emphasized. Based on this study, authors concluded that there exist several opportunities for energy savings in SSL. These opportunities mainly emerge from the operating flexibility provided by SSL, and the availability of various parameters in cryptographic algorithms like key size and number of rounds that can be tuned to attain the best trade-off between the security level and energy consumption, as described earlier in the previous section.

Nevertheless, the reliance of SSL on conventional public key cryptography like RSA for shared secret key exchange makes it too expensive and impractical for highly constrained embedded devices like sensor motes. Based on this observation, Gupta et al. [45] propose the use of elliptic curve cryptography in SSL, in order to enhance the protocol performance in low-end embedded systems. For that, two ECC-based cryptographic primitives namely ECDH and ECDSA have been incorporated in the SSL's cipher suite in order to be used respectively in key exchange and authentication. Moreover some modifications have been introduced on the protocol's behaviour; for example, only a single cipher suite (ECC-based) has been enabled, and both certificate and session's identifier sizes have been reduced. As a result, it has been shown through real implementations that theses modifications allow the creation of an entire secure web server stack, including SSL, that runs efficiently within mica2 [89] sensor motes.

A similar study has been conducted by karri et al. [30] to optimize the energy consumed by the network-layer security protocol IPSec [55] in wireless networks. Based on test bed measurements, authors identified the various sources of energy consumption during a wireless IPSec session. Then, they suggest several optimization techniques based on exchanged data compression, adaptive session negotiation, and cryptographic primitives selection in order to reduce the IPSec energy dissipation. However, we note that such studies only reflect the energy behavior of the studied protocol on a specific embedded device, and practically no result can be derived for other security protocols running on different embedded systems. We think that much more efforts need to be devoted for the energy consumption modeling of the security protocol's building blocs and their constituent elementary operations with the respect to various embedded architectures. This will better help designers to project the energy requirements of a large variety of security protocols for different embedded architectures.

Moreover, we remark that the proposed optimization techniques like adaptive session negotiations and certificate's size reductions highly depend on the protocol's functioning and do not apply for all security protocols. Consequently, it is important to investigate the development of protocol-level generic optimization methods that can be integrated in every security protocol to enhance its energy consumption. For example, Yuan et al [32] consider the insertion of additional information into the protocol transmitted message in order to guide the selection of proper processor's voltage needed for their processing (e.g., decryption of transmitted data). For that, Dynamical Voltage Scaling (DVS) technique is used to allow embedded processors to operate at an energy-efficient manner by dynamically adapting voltage and therefore the clock frequency, for data encryption/decryption and processing phases. Based on the assumption that energy requirement for encryption/decryption in asymmetric ciphers is not proportional to the message length, authors presume that the introduction of additional header in the trans-

mitted message enhance the energy expenditure for its encryption/decryption at the receiver. However, we remark that the energy gained at the receiver may not compensate the additional energy consumed by the sender during the added header transmission.

E. Secure embedded networking

It has been stressed earlier that the insecure nature of the wireless medium, mostly used in embedded networks, mandates the resort to link layer security measures that become significantly more critical than in wired networks. In this section, we outline the different embedded networks technologies and standards that rely on link layer security mechanisms and discuss their underlying security architectures. Further, we also identify the most prominent security concerns that are typical to a specific embedded network's application, and we overview their corresponding mitigation techniques.

E.1 Link-layer security protocols

Apart the weak security of the wireless communication medium [112], it may exist other causes that orient network designers toward link-layer security solutions. In fact, ensuring end-to-end communication security, through transport-layer security protocols for example, does not constitute a viable solution in certain application scenarios. For instance; in sensor networks, the communication paradigm is based on in-network processing that allows elimination of redundant sensory data and aggregation of reported readings during their routing from the sensing nodes to the base station. Hence, intermediate routers need to access, modify and suppress the content of transmitted messages making it impossible the use of end-to-end security mechanisms between each sensor and the base station. In such situation, link layer security mechanisms, that provide communication security during each elementary transmission between neighboring nodes, represents the adequate solution.

Hereafter, we invoke different types of embedded communication networks and standards that privilege link-layer security. By the given overview, we intend to highlight the open communication security problems, which are in most cases induced by the resource requirement versus security level trade-off.

1. *GSM:* Global System of Mobile communications (GSM) is considered as the most widely used cellular network technology for mobile phone services. It incorporates several security mechanisms that protect the network by authenticating customers, and provide privacy for the users by encrypting their conversations while transmitted over the air. To ensure the voice data confidentiality, the stream-cipher A5 is used to encrypt GSM frames. Although this algorithm was never made public; it has been reverse-engineered and cryptanalyzed. Unfortunately, the cryptanalysis [46] found that some A5 versions, most notably A5/2, contain serious vulnerabilities that allow attackers to compromise data confidentiality in a short time, and recover the initial secret encryption key using ciphertext attack. We also mention that no mechanism was integrated in GSM security specifications to ensure data integrity.
2. *IEEE 802.11:* The 802.11 wireless communication standard also integrates several mechanisms to ensure security in wireless local area networks. It initially specified WEP, a scheme that uses RC4 encryption for confidentiality, and a CRC checksum for integrity protection. In addition, a shared key authentication is utilized to allow mobile stations authentication through the use of a standard challenge/response protocol along with a shared WEP secret key. However, security researchers quickly found that WEP presents several design security flaws that consist essentially in (i) the low size of the 24-bit adopted IVs (initialization vectors), (ii) the CRC checksum inefficiency to protect data integrity, and (iii) the nave use of IVs to diversify RC4's keys which allows devastating cryptanalytic key-related attacks [47–49]. The identified WEP deficiencies led also to the compromise of the shared key authentication scheme. Subsequently, the 802.11 design group proposed the 802.11i's CCMP (Counter-mode/CBC Mac Protocol), which ensures security by using AES in CCM mode for encryption, along with 48-bit IVs and a strong message authentication code. Although that CCMP appears to be well designed, it involves a substantial per-packet overhead compared to WEP, moreover it requires WEP users to upgrade the hardware in order to enable strongest security mechanisms.
3. *Bluetooth:* Bluetooth is another standard defined by the Special Interest Group (SIG) to enable communication between embedded devices in a peer-to-peer wireless network. Bluetooth specifications adopted some cryptographic link-layer mechanisms notably, a 128-bit secret key authentication technique, and E0 stream cipher for sensitive data encryption. However, this latter has also been shown to be flawed, and the overcome to certain attacks presented in [50, 51] remains still an open topic.
4. *Sensor networks:* Wireless Sensor Networks (WSNs) represent an important application of embedded networking that involves one of the most resources constrained devices, namely wireless micro-sensors. These latter have a very limited capability in term of available energy, memory capacity and processing speed. They vary from those called smart dust [52] with only 8 KB of program and 512 bytes of data memory, and processors with 32 8-bit general registers that run at 4MHZ and 3.0V, to sensors like mica2 [89] that are over an order of magnitude more capable in term of processing speed (using ATMEL ATmega128L), and memory size (128 KB of program flash memory).
   As it has been emphasized earlier, the communication model of WSNs mainly characterised by a many-to-one traffic pattern and in-network processing necessitate the resort to link-layer security mechanisms. Con-

sequently, almost all the proposed security protocols for sensor networks fit in this category.
SPINS [107] is considered as one of the first proposed security schemes in WSNs. It consists of two building blocs namely SNEP and TESLA. SNEP provides data confidentiality via chaining encryption function (e.g., RC5-CBC), it also employs a shared counter between every two communicating nodes for preventing replay attacks and ensuring data freshness. SNEP uses a message authentication code to guarantee two-party authentication and data integrity. On the other hand, TESLA focus on the problem of providing efficient broadcast authentication, which is a very important issue in the context of sensor networks. For that, the protocol avoids the use of asymmetric cryptography by introducing asymmetry through a delayed disclosure of symmetric MAC keys. Nevertheless, SPINS architecture relies on the concept, that every node shares a secret key with a trusted base station, which is always able to communicate with every node in the network. Moreover, it requires a loose synchronization between communicating nodes.
Further, there exist in the literature several security platforms [53,94] that demonstrate that software-only solutions are indeed practical with today's sensor technology and hardware support like that used in 802.15.4 standard [54] is not needed to achieve acceptable security and performance levels. For instance, the University of California, Berkeley, implementation of TinySec [53] incurs only an additional 5%-10% performance overhead using software-only methods.

E.2 Typical networking threats prevention

Embedded networks cover a broad range of actual and emerging networks, such as: MANETs (Mobile Adhoc NETworks), WSNs (Wireless Sensor Networks), IVCNs (Inter-Vehicle Communication Networks), and so fourth. Many solutions have been proposed to overcome typical security threats against this kind of networks. Indeed, embedded networks incarnate typical features that raise serious security attacks: the resource constrained nature of embedded devices, drives embedded network nodes to get away from taking part in networking services in order to save their resources. This is commonly called "selfishness", and a couple of solutions have been proposed to tackle this misbehavior. The infrastructure-less and ubiquitous communication nature of embedded networking raises serious security weaknesses with respect to "trustiness" and "secure data dissemination". Another typical characteristic of embedded networks, such as IVCNs, is localization capability using GPS (for instance). This useful functionality for some applications threatens the privacy of the mobile users (drivers and passengers in case of IVCNs). Therefore, many solutions have been proposed to guarantee "anonymity" for localization enabled embedded devices. In what follows, we present some solutions dealing with these four security services: selfishness avoidance, trustiness establishment, secure data dissemination, and anonymity.

1. *Selfishness avoidance:* There are two main modes for selfishness avoidance: the detection mode and the preventive mode. In the detection mode, many solutions have been proposed to detect misbehaving nodes: Papadimitratos and Haas [56], and Awerbuch et al. [57] proposed to use End-to-End feedbacks, where a successful acknowledgment implies that the corresponding route is operational, while a failure in the acknowledgments may be considered as an indication that the route is broken, compromised, or includes selfish nodes. Marti et al. [58], and Kargi et al. [59]proposed mechanisms relying on the watchdog method that aims to detect misbehaving nodes that do not forward packets: a node rates its neighbors depending on whether they forward or not received packets.
In the preventive mode, Buttyan and Hubaux [60,61] have proposed an economic based approach that stimulates the nodes to cooperate. The main idea of this technique is that nodes which use a service must pay for it (using nuglets) to nodes that provide the service. This makes nuglets indispensable for using the network, and motivates each node to increase its stock of nuglets by providing services to other nodes.
2. *Trustiness establishment:* Trustiness is a compulsory security service in infrastructure-less and ubiquitous embedded networks. Many cryptographic and behavior-based solutions have been proposed. Cryptographic solutions consist of using authentication techniques during all routing phases, to exclude attackers and unauthorized nodes from participating in the routing. Most of the proposed solutions belonging to this class modify existing routing protocols to build authentication-based solutions [57,62–64].
Behavior-based solutions rely on the analysis of nodes' behavior to attribute a trust level to each of them to be considered in routing and data dissemination services. Yi et al. [65] proposed to use a hierarchical trust values metric and authentication. To define nodes' trust values, the authors address the example of a military context, in which the trust level matches to node's owner rank. Hu et al. [66] proposed to use a challenge-response technique to determine whether a neighbor could be trusted. Yang et al. [67] have proposed a mutually according admission approach: nodes in a neighborhood collaboratively monitor each other to detect any misbehavior. Then nodes mutually accord participation admissions, and nodes without up-to-date admissions are excluded from any network service. Michiardi and Molva [68] proposed a solution that relies on a dynamic trust value attribution by propagating positive observations of well-behaving nodes. On the other hand, Buchegger and LeBoudec proposed in [70,71] a dynamic trust value attribution solution which is built on negative experiences rather than positive impressions. Buchegger et al. [69] justified this by the fact that misbehaving is the exception rather than the norm.

3. *Secure data dissemination:* MANETs and ubiquitous networks are hostile environments where a special attention must be given to guarantee packet delivery and data dissemination. Many proposed solutions rely on introducing redundancy to provide reliability for data dissemination: Papadimitratos and J. Hass [56] have proposed a solution based on Rabin's algorithm [106], which takes advantage of the existence of multiple routes from a source to a destination to increase reliability when transmitting packets. It consists of adding redundancy to the message to send, then the message and the redundancy are encoded and divided into a number of pieces and dispersed on the available routes, so that even a partial reception can lead to the successful reconstruction of the message at the receiver. This technique can mitigate partial packet loss that can occur due to misbehavior on some used routes, since the encoding and the dispersion allows the reconstruction of the original message with successful reception of any M out of N transmitted pieces.
   Another similar security concern is how to guarantee data aggregation, which is a key emerging theme in the design and development of WSNs. In this process, intermediary nodes called "aggregators" collect the raw sensed information from sensor nodes, process it locally, and forward only the result to base stations. Possible threats against this hierarchical computation model can vary from denial-of-service attacks that try to stop completely this service to stealthy attacks where the attacker's purpose is to make the user accept false aggregation results. For data aggregation validity assurance Du et al. [72] have proposed the use of redundant data fusion nodes as witnesses. These nodes conduct the same data fusion operations as aggregators, but send the result as a Message Authentication Code (MAC) to the aggregator itself instead of sending it to the base station. In order to prove the validity of the aggregation results, the aggregator has to forward the received proofs from witness nodes along with its calculated result to the base station. If a compromised aggregator wants to send invalid fusion data, it has to forge the proofs on the invalid results. The aggregation result is confirmed when M out of N witness proofs agree with the aggregators' results, otherwise this latter is discarded and the base station polls one of the witness node to send it the valid aggregation result.
4. *Anonimity:* One of the security issues in positioning-enabled equipments, such as vehicles in IVCNs, is anonymity. Indeed, a secure inter-vehicle communication system should be able to help establish the liability of drivers; but at the same time, it should protect as far as possible the privacy of the drivers and passengers [74, 76]. Raya and Hubaux [73, 75] proposed a general PKI-based architecture to secure inter-vehicle communications. In the proposed solution, an attacker can rely on key correlation to track victim vehicles and hence harm passengers' privacy. To guarantee anonymity, the authors proposed in [75] a key changing algorithm that adapts to the vehicle speed and takes into account key correlation by the attacker. It consists in limiting the life time of used keys by a vehicle depending on its speed in order to break any correlation between the identity of a vehicle and used keys and hence guarantee anonymity.

V. Conclusion

In this paper we have discussed the most important issues concerning security in embedded systems. We have shown that addressing these issues proves to be a challenging task, wherein embedded devices need to be secure to the extent required by the application and the environment without compromising performance, energy consumption, cost and usability.

Through the survey of existing security solutions, we noticed that securing embedded systems opens a large number of contribution areas and research fields. In fact, the need to consider security objectives as mainstream design factors of embedded systems requires more extensive investigations on the methods allowing a deeper waving of security considerations in the system development life cycle. We believe that this objective imperatively passes through the development of a more comprehensive threat models and efficient testing methods.

Another contribution area in the embedded security world consists of implementing a more effective security processing architectures that achieve a better hardware and software combinations with enhanced flexibility and a lower cost. For that, an interdisciplinary endeavour is required to attain powerful hardware operations and optimized software processing. Concerning the software optimization side, we think that research efforts should be intensively oriented to the energy consumption efficiency. Indeed, the Moor's law confirms that the processing performance is growing exponentially, while we notice only a linear increase of batteries' capacity. This fact mandates a special care of the energy efficiency factor amongst other performance concerns in order to reduce the gap between security computing requirements and batteries' supply.

And finally, we showed that the close physical coupling of embedded system with their specific operating environment require secure architecture designers to look beyond the basic security goals and provide defences against broad classes of specific attacks including those related to the embedded networking applications. In some of these applications, like sensor networks, we realize that security design remains a field in its infancy, and presents an attractive area of contributions.

References

[1] P. Kocher, J. Jaffe,B. Jun. Using unpredictable information to minimize leakage from smartcards and other cryptosystems. USA patent, International Publication number WO 99/63696,1999.
[2] S. Fruhauf,L. Sourge. Safety device against the unauthorized detection of protected data. U.S. patent 5,404,402,1995.
[3] J. J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," Lec-

ture Notes in Computer Science (Smartcard Programming and Security), vol. 2140, pp. 200-210, 2001.
[4] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers," IEEE Trans. Computer-Aided Design, vol. 21, pp. 1509-1517, Dec. 2002.
[5] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater, "On a new way to read data from memory," in Proc. First Intl. IEEE Security in Storage Wkshp, pp. 65-69, Dec. 2002.
[6] CryptocellTM. Discretix Technologies Ltd. (http://www.discretix.com).
[7] B. Yee, Using Secure Co-processors. PhD thesis, Carnegie Mellon University, 1994.
[8] Secure Coprocessing. IBM Inc, http://www.research.ibm.com/scop/
[9] A. Arbaugh, D. J. Farber, and J. M. Smith, "A Secure and Reliable BootStrap Architecture," in Proc. of IEEE Symposium on Security and Privacy, pp. 65-71, May 1997.
[10] Y. Chen, R. Venkatesan, M. Cary, S. Sinha, and M. H. Jakubowski, "Oblivious hashing: A stealthy software integrity verification primitive," in Proc. Int. Wkshp. Information Hiding, pp. 400-414, Oct. 2002.
[11] G. C. Necula and P. Lee, "Proof-Carrying Code," Tech. Rep. CMU-CS-96-165, Carnegie Mellon University, Nov. 1996.
[12] C. Collberg, C. Thomborson, and D. Low, "Breaking Abstractions and Unstructuring Data Structures," Proc. IEEE Int'l Conf. Computer Languages (ICCL '98), May 1998.
[13] Trusted Computing Group. (https://www.trustedcomputinggroup.org/home).
[14] Next-Generation Secure Computing Base (NGSCB). Microsoft Inc. (http://www.microsoft.com/resources/ngscb/productinfo.mspx).
[15] N. Daswani, D. Boneh, "Experimenting with Electronic Commerce on the PalmPilot", Lecture Notes in Computer Science, vol. 1648, 1-16, 1999.
[16] J. Park, J. Hwang, Young-Chul Kim, "FPGA and ASIC Implementation of ECC Processor for Security on Medical Embedded System", Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), 2005.
[17] Safenet EmbeddedIPTM. Safenet Inc. (http://www.safenet-inc.com).
[18] S. Ravi, A. Raghunathan, N. Potlapally, and M. Shankaradass, "System design methodologies for wireless security processing platform," in Proc. Design Automation Conf., pp. 777-782, June 2002.
[19] R. B. Lee, Z. Shi, and X. Yang, "Efficient permutations for fast software cryptography," IEEE Micro, vol. 21, pp. 56-69, Dec. 2001.
[20] Z. Shi, X. Yang, and R. B. Lee, "Arbitrary bit permutations in one or two cycles," in Proc. Int. Conf on Application-Specific Systems, Architectures and Processors, pp. 237-247, June 2003.
[21] R. B. Lee, Z. Shi, and X. Yang, "How a processor can permute n bits in O(1) cycles," in Proc. Hot Chips 14 - A Symposium on High Performance Chips, Aug. 2002.
[22] K. Araki, T. Satoh, and S. Miura, "Overview of Elliptic Curve Cryptography," Springer-Verlag Lecture Notes in Computer Science, vol. 1431, pp. 29-48, 1998.
[23] E. Savas, A. F. Tenca, and C. K. Koc, "A Scalable and Unified Multiplier Architecture for Finite Fields GF(p) and GF(2n)," Springer-Verlag Lecture Notes in Computer Science, vol. 1965, pp. 277-292, 2000.
[24] A. M. Fiskiran and R. B. Lee, PAX: A Datapath-Scalable Minimalist Cryptographic Processor for Mobile Environments (in Embedded Cryptographic Hardware: Design and Security). Nova Science Publishers (to be published), 2004.
[25] HIFN Inc. http://www.hifn.com.
[26] Point-to-Point Protocol (PPP), RFC 1661. The Internet Engineering Task Force. Available at http://www.ietf.org/rfc/rfc1661.
[27] Point-to-Point Tunneling Protocol (PPTP), RFC 2637. The Internet Engineering Task Force. Available at http://www.ietf.org/rfc/rfc2637.
[28] James Ross Goodman, "Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications", PHD Thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, August 2000.
[29] A. Hodjat and I. Verbauwhede. The energy cost of secrets in ad-hoc networks. In Proceedings of the IEEE CASWorkshop on Wireless Communications and Networking. IEEE, 2002.
[30] R. Karri and P. Mishra, "Optimizing IPSec for energy-efficient secure wireless systems," System-level Power Optimization for Wireless Multimedia Communication, Kluwer Academic Publishers, pp. 133-152, ISBN: 1-4020-7204-X
[31] G. Gaubatz, J.-P. Kaps, E. ztrk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks", Workshop on Pervasiv Computing and Communications Security - PerSec'05, IEEE Computer Society, pages 146-150, March, 2005.
[32] Lin Yuan, Gang Qu. Design Space Exploration for Energy-Efficient Secure Sensor Network. The IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP'02) July 17 - 19, 2002 San Jose, California
[33] C. T. Hager, S. F. Midkiff, J.-M. Park, and T. L. Martin, "Performance and Energy Efficiency of Block Ciphers in Personal Digital Assistants," IEEE International Conference on Pervasive Computing and Communications (PerCom), March 8-12, 2005.
[34] Prasanth Ganesan , Ramnath Venugopalan , Pushkin Peddabachagari , Alexander Dean , Frank Mueller , Mihail Sichitiu, Analyzing and modeling encryption overhead for sensor network nodes, Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, September 19-19, 2003, San Diego, CA, USA.
[35] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, Improved Cryptanalysis of Rijndael, Fast Software Encryption, 2000 pp213-230.
[36] National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information, CNSS Policy No. 15, Fact Sheet No. 1, June 2003.
[37] Joan Daemen and Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer-Verlag, 2002. ISBN 3540425802
[38] B. Schneier, "Applied Cryptography", Second edition, John Wiley & Sons, 1996.
[39] W. Freeman, E. Miller, "Experimental analysis of cryptographic overhead in performance-critical systems", 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 1999, pp. 348-357.
[40] N. Potlapally, S. Ravi, A. Raghunathan, and G. Lakshminarayana, "Optimizing Public-Key Encryption for Wireless Clients," in Proc. IEEE Int. Conf. Communications, pp. 1050–1056, May 2002.
[41] J. J. Quisquater and C. Couvreur, "Fast Decipherment algorithm for RSA public-key cryptosystems," in Electronic Letters, pp. 905-907, Oct. 1982.
[42] P. L. Montgomery, "Modular multiplication without trial division," in Mathematics of Computation, pp. 519-521, 1985.
[43] Sun Tzu, "Sun Tzu Suan Ching", third century CE.
[44] Sohail Hirani, "Energy Consumption of Encryption Schemes in Wireless Devices", MS thesis, University of Pittsburgh 2003.
[45] V. Gupta et al., "Sizzle: A standards-based end-to-end security architecture for the embedded Internet", 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), 8-12 March 2005, Kauai Island, HI, USA
[46] Elad Barkan, Eli Biham, and Nathan Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communication. In Advances in Cryptology - CRYPTO 2003, volume 2729 of Lecture Notes in Computer Science, 2003.
[47] Adam Stubbleld, John Ioannidis, and Aviel D. Rubin. Using the uhrer, mantin, and shamir attack to break WEP. In Network and Distributed Systems Security Symposium (NDSS), 2002.
[48] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: The insecurity of 802.11. In The Seventh Annual International Conference on Mobile Computing and Networking (MobiCom 2001), 2001.
[49] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. Lecture Notes in Computer Science, 2259:1-24, 2001.
[50] Yi Lu and Serge Vaudenay, Cryptanalysis of Bluetooth Keystream Generator Two-Level E0, ASIACRYPT 2004, pp483-499.
[51] Markus Jakobsson and Susanne Wetzel. Security weaknesses in Bluetooth. In CT-RSA 2001, pages 176-191. Springer-Verlag, 2001. LNCS 2020.
[52] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for "Smart Dust,"" Int'l. Conf. Mobile Computing and Net. (MOBICOM), 1999, pp. 271-78.

[53] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", ACM SenSys, Nov. 2004.

[54] Naveen Sastry and David Wagner. Security considerations for IEEE 802.15.4 networks. In ACM Workshop on Wireless Security (WiSe 2004), October 2004.

[55] KENT, S. AND ATKINSON, R. 1998. RFC 2401: Security Architecture for the Internet Protocol. Corporation for National Research Initiatives, Internet Engineering Task Force, Network Working Group, Reston, Virginia, USA.

[56] P. Papadimitratos and Z. J. Haas, "Secure Data Transmission in Mobile Ad Hoc Networks," 2003 ACM Wksp. Wireless Security, San Diego, CA, USA, 2003, pp. 41-50.

[57] B. Awerbuch et al., "An On Demand Secure Routing Protocol Resilient to Byzantine Failures," ACM Wksp. Wireless Security (WiSe), Atlanta, Georgia, Sept. 2002.

[58] S. Marti et al., "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," ACM Mobile Comp. and Net., MOBICOM 2000, pp. 255-65.

[59] F. Kargl et al., "Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks," 1st European Wksp. Security in Ad-Hoc and Sensor Networks, ESAS 2004, Aug. 5-6, 2004.

[60] L. Buttyan and J.-P. Hubaux, "Nuglets: A Virtual Currency to Stimulate Cooperation in Self-organized Mobile Ad Hoc Networks," Swiss Federal Institution of Technology, Lausanne, Switzerland, Tech. Rep. DSC/2001/001, Jan. 2001.

[61] L. Buttyan and J.-P. Hubaux, "Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks," ACM/Kluwer Mobile Net. and Applications (MONET), vol. 8, no. 5, Oct. 2003.

[62] Y.C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks," 8th Annual Int'l. Conf. Mobile Comp. and Net. MobiCom 2002, Sept. 2002, pp. 12-23.

[63] C. Castelluccia and G. Montenegro, "Protecting AODV Against Impersonation Attacks," ACM SIGMOBILE Mobile Comp. and Commun. Rev. Archive, vol. 6, no. 3, July 2002, pp. 108-09.

[64] K. Sanzgiri et al., "A Secure Routing Protocol for Ad Hoc Networks," 10th IEEE Int'l. Conf. Network Protocols (ICNP '02), Nov. 2002.

[65] S. Yi, R Naldurg, and R. Kravets, "Security-aware Ad-hoc Routing for Wireless Networks," ACM Wksp. Mobile Ad Hoc Networks, Mobihoc, 2001.

[66] Y.C. Hu, A. Perrig, and D. B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," ACM Wksp. Wireless Security WiSe 2003, San Diego, CA, USA, Sept. 2003.

[67] H. Yang, X. Meng, and S. Lu, "Self-organized Network Layer Security in Mobile Ad Hoc Networks," ACM MOBICOM Wireless Security Wksp (WiSe'02), Sept. 2002.

[68] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," Commun. and Multimedia Security 2002 Conf., Portoroz, Slovenia, Sept. 26-27 2002.

[69] S. Buchegger and J.-Y. LeBoudec, "A Robust Reputation System for Mobile Ad-hoc Networks," EPFL IC, Tech. Rep. IC/2003/50, July 2003.

[70] S. Buchegger and J. LeBoudec, "Performance Analysis of the Confidant, Protocol Cooperation of Nodes Fairness in Dynamic Ad Hoc Networks," 3rd ACM Int'l. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc '02), Lausanne, Switzerland, June 2002, pp. 80-91.

[71] S. Buchegger and J.-Y. Le-Boudec, "A Robust Reputation System for p2p and Mobile Ad-hoc Networks," 2nd Wksp. Economics of Peer-to-Peer Systems, June 2004.

[72] W. Du et al., "A Witness-based Approach for Data Fusion Assurance in Wireless Sensor Networks," GLOBECOM '03, 2003.

[73] Raya, M. and Hubaux, J.-P. The security of vehicular networks Technical report, EPFL, 2005.

[74] Maxim Raya and Jean-Pierre Hubaux, Security Aspects of Inter-Vehicle Communications, STRC 2005 (Swiss Transport Research Conference), March 2005

[75] Maxim Raya and Jean-Pierre Hubaux, The Security of Vehicular Ad Hoc Networks, In SASN'05, November 2005.

[76] Jean-Pierre Hubaux, Srdjan Capkun and Jun Luo, The Security and Privacy of Smart Vehicles, IEEE Security & Privacy Magazine, Vol. 2, No. 3, pp 49–55, May-June 2004.

[77] Bruce Schneier, "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)," The Cambridge Algorithms Workshop, LNCS(809),1994.

[78] N. Xu, "A Survey of Sensor Network Applications," IEEE Communications Magazine, 40(8):102-114, August 2002.

[79] Zakaria Maamar: Commerce, e-commerce, and m-commerce: what comes next? Commun. ACM 46(12): 251-257 (2003)

[80] ePaynews - Mobile Commerce Statistics. http://www.epaynews.com/statistics/mcommstats.html

[81] I.F.Akyildiz,W.Su ,Y.Sankarasubramaniam, E.Cayirci, "Wireless sensor networks: a survey" , Computer Networks: The International Journal of Computer and Telecommunications Networking, v.38 n.4, pp.393-422, 2002

[82] S. Ravi, A. Raghunathan, P. Kocher and S. Hattangady, "Security in Embedded Systems: Design Challenges" in ACM Transactions on Embedded Computing Systems: Special Issue on Embedded Systems and Security (Guest Editors: D. Serpanos and H.Lekatsas), 2004

[83] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as A New Dimension in Embedded System Design," in Proc. Design Automation Conf., June 2004, pp. 753-760.

[84] S. Ravi and A. Raghunathan and N. Potlapally,"Securing Wireless Data: System Architecture Challenges",in Proc. Int. Conf. on System Synthesis,October,2002.

[85] OpenSSL, "http://www.openssl.org".

[86] R. Ernst,"Codesign of Embedded Systems: Status and Trends", IEEE Design & Test of Computers,15(2),1998

[87] G. De Micheli and R.K. Gupta, "Hardware-Software Codesign", IEEE Design & Test of Computers,85(3),1997.

[88] Y. Matsuoka and P. Schaumont and K. Tiri and I. Verbauwhede, "Java cryptography on KVM and its performance and security optimization using HW/SW co-design techniques", Proc. Int. Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2004), 303-311, September, 2004.

[89] Crossbow Co., "MICA, MICA2 Motes & Sensors", http://www.xbow.com/, 2005.

[90] Nachiketh R. Potlapally and Srivaths Ravi and Raghunathan and Niraj K. Jha, "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols", IEEE Transactions in Mobile Computing, 5(2), February, 2006.

[91] S. Ravi and A. Raghunathan and S. Chakradhar, "Tamper Resistance Mechanisms for Secure Embedded Systems", IEEE Intl. Conf. on VLSI Design, January, 2004.

[92] R. Anderson and M. Bond and J. Clulow and S. Skorobogatov, "Cryptographic Processors - A Survey", Technical Report UCAM-CL-TR-641, University of Cambridge, August, 2005.

[93] YongBin Zhou and DengGuo Feng, "Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing", NIST Physical Security Testing Workshop, September, 2005.

[94] D. Djenouri and L. Khelladi and N. Badache, "A survey of security issues in mobile Ad hoc and sensor networks", IEEE communications surveys, 7(4), December, 2005.

[95] W. A. Arbaugh and N. Shankar and J. Wang, "Your 802.11 network has no clothes", The First IEEE International Conference on Wireless LANs and Home Networks, 131-144, December, 2001.

[96] FIPS PUB 140-2, "Security Requirements for Cryptographic Modules", http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

[97] Tim Grance and Joan Hash and Marc Stevens, "Security Consideration in the Information System Development Life Cycle", NIST, October, 2003.

[98] R. L. Jones and A. Rastogi, "Secure Coding: Building Security into the Software Development Life Cycle", Information Systems Security, November, 13(5), 2004.

[99] J. ZAMBRENO and A. CHOUDHARY and R. SIMHA and B. NARAHARI and N. MEMON, "SAFE-OPS: A Compiler/Architecture Approach to Embedded Software Security", ACM Trans. Embedded Computing, 4(1), February, 2005.

[100] Larus J. R. and Ball T. and Das M. and DeLine R. and Fhndrich M. and Pincus J. and Rajamani S. K. and and Venkatapathy R., "Righting Software", IEEE Software, 21(3), 92-100, May-June, 2004.

[101] Wang and J. Hill and J. Knight and J. Davidson, "Software Tamper Resistance: Obstructing Static Analysis of Programs", Technical Report, Univ. of Virginia, 2000.

[102] "MJ Atallah and ED Bryant and MR Stytz, "A survey of anti-

tamper technologies", CROSSTALK The Journal of Defense Software Engineering, November, 2004.
[103] K. Okeya and T. Takagi, "A More Flexible Countermeasure against Side Channel Attacks Using Window Method", CHES'2003, LNCS(2779), 397-410, 2003.
[104] S. Chari and C. Jutla and J. Rao and P. Rohatgi,"Towards sound approaches to counteract power-analysis attacks", CRYPTO'99, LNCS(1666), 398-412, 1999.
[105] L. Goubin, "A sound method for switching between boolean and arithmetic masking", CHES 2001, LNCS(2162), 3-15, 2001.
[106] M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance", Journal of the Association for Computing Machinery, 36(2), 335-348, April, 1989.
[107] Adrian Perrig and Robert Szewczyk and J.D. Tygar and Victor Wen and David E. Culler,"SPINS: Security Protocols for Sensor Networks", Wireless Networks, Kluwer Academic Publishers, 8, 521-534, 2002.
[108] Federal Information Processing Standards Publication, "Data Encryption Standard (DES)", FIPS PUB 46, December, 1993.
[109] Federal Information Processing Standards Publication, "Advanced Encryption Standard (AES)", FIPS PUB 197, November, 2001.
[110] Federal Information Processing Standards Publication, "Digital Signature Standard (DSS)", FIPS PUB 186, May, 1994.
[111] W. Diffie and M. E. Hellman, "New Directions in Cryptography", IEEE Trans. Info. Theory, IT-22, 644-654, November, 1976.
[112] Mohamed G. Gouda, E.N. Elnozahy, Chin-Tser Huang, and Tommy M. McGuire. Hop integrity in computer networks. IEEE/ACM Transactions on Networking, 10(3):308-319, June 2002.
[113] Compaq Ipaq Pocket PC. http://h20022.www2.hp.com, 2002.