Journal of Information Assurance and Security
ISSN 1554-1010 Volume 6 (2011) pp. XXX-XXX

A Secure Architecture for Re-Taskable Sensing Systems

Alessandra De Benedictis[1], Andrea Gaglione[1] and Nicola Mazzocca[1]

[1]University of Naples Federico II, Department of Computer Science and Systems,
Via Claudio, 21 80125 Napoli, Italy
{alessandra.debenedictis, andrea.gaglione, nicola.mazzocca}@unina.it

***Abstract*: Sensor Networks are considered a high-innovation potential branch in the field of network computing and are widely used in several application domains thanks to their cost effectiveness, flexibility and ease of deployment. They are well suited to a multitude of monitoring and surveillance applications and are often involved in mission-critical tasks, thus making security a primary concern. Many architectures and protocols have been proposed to address this issue, mainly based on cryptographic operations, but it still represents an open research area: in fact, in order to be effective, such techniques often require complex computations and a large amount of dedicated resources, which are not available on sensor platforms according to the existing technology. Nevertheless, if considering tiered sensor networks, where tiny motes coexist with more powerful nodes, it is possible to perform some complex and efficient security schemes by exploiting the different capabilities of nodes. In this paper we present a secure architectural proposal based on the Tenet system, a tiered re-taskable sensor network architecture. Specifically, we have integrated security features into the Tenet architecture in order to implement a hybrid cryptosystem. Such a cryptosystem combines symmetric and asymmetric cryptographic schemes to benefit of the security provided by asymmetric protocols and the better performance of symmetric ones.**

***Keywords*:** *Sensor network security, Secure communication architecture, Tiered sensor networks.*

I. Introduction

The increasing spread of sensor networks has led to the diffusion of middleware platforms as well as sensor network programming systems, aiming to bridge the gap between applications and the underlying hardware platforms. These systems provide high level programming abstractions and implement services such as routing, transport, task dissemination and execution and time synchronization, thus simplifying application development. Tenet [1], [2] is an example of such systems and the validity of its architecture has been demonstrated in several application domains [3], [4].

The Tenet architecture has been conceived for tiered sensor networks consisting of two classes of devices: the lower tier is composed of small-form-factor, resource-poor nodes, named motes, which enable flexible deployment of dense instrumentation, while in the upper tier less constrained 32-bit nodes (named masters) implement multi-node data fusion and application logic.

One of the main open issues in such kind of systems is related to the development of general purpose security protocols. There is a large number of application scenarios where data exchanged between sensor nodes is critical (e.g. health or military applications), and providing security services for such applications is a technical challenge, due to hostile deployment environments and resource limitations.

The openness of wireless channels lets anyone be able to sniff or participate in communications, undermining integrity and confidentiality requirements of the system; moreover, unattended physical access to the network infrastructure may encourage node capture and redeployment or even the placement of malicious nodes into the network causing an unattended behavior (i.e. redirecting or interrupting communication and service).

These specific challenges, together with the limited energy, computation, and communication capabilities of sensor devices, make it difficult to directly employ the existing security approaches to the area of wireless sensor networks.

Most security protocols are based on cryptographic operations as encryption and authentication; they massively involve the adoption of keys and complex mathematical functions that require dedicated computational resources. Indeed, the adoption of such security mechanisms on so small devices can be critical from a performance and power consumption point of view. At this aim, in this paper, we discuss the design and implementation of a hybrid cryptosystem that combines symmetric and asymmetric cryptographic schemes, in order to benefit from both the higher level of security provided by asymmetric protocols and the better performance of symmetric ones.

Such a cryptosystem can be more effectively implemented in a tiered system like Tenet, where the advantage of having different computational and energy constraints between the motes and the base station can be exploited. As a matter of fact, tiers are not only a fundamental condition to scale network size and spatial extent, as the higher level nodes have greater network capacity and larger spatial reach than a flat network, but they also allow a computational load partition between nodes in such a way that "master" nodes can perform more complex cryptographic operations without affecting the performance of the overall system.

The reminder of the paper is structured as follows: in Section 2 a description of security issues in sensor networks is provided, along with the discussion of the main solutions known in literature. In Section 3 we will give a brief overview of the Tenet architecture, while in Section 4 we will illustrate our proposal as well as our aims. In Section 5 we will describe our security enhanced Tenet architecture and also give some implementation details; finally, in Section 6 some conclusions and future work will be drawn.

II. Security issues in Sensor Networks and state of the art

Recently there has been an intense research aimed at developing security schemes for sensor network applications, as they are well suited to a multitude of monitoring and surveillance applications and are often involved in mission-critical tasks, thus making security a primary concern. In this section, we will discuss about the main security issues in WSNs, presenting an overview of the state of the art and focusing on the aspects that motivated our proposal.

In [5] the authors identify and summarize the main threats to Wireless Sensor Networks (WSN) and their vulnerabilities, and give a brief summary of security issues and defense suggestions from the point of view of the OSI model.

On the basis of the Dolev-Yao threat model [6], an attacker can spoof, intercept, alter and inject any message exchanged between sensor nodes; due to the features of WSN in fact, there are some specific attacks targeting the communication channels: an adversary can easily retrieve valuable data from the transmitted packets that are sent (eavesdropping), simply intercept and modify the packets' content meant for the base station or intermediate nodes (message modification), or re-transmit the contents of those packets at a later time (message replay); finally, he can send out false data into the network, maybe masquerading as one of the sensors, with the objectives of corrupting the collected sensors' reading or disrupting the internal control data (message injection).

According to that, main requirements of secure sensor network architectures are *authentication*, *confidentiality*, *integrity*, and *freshness*, meant as the property of exchanged data to be recent, that is not replayed by an adversary from an old message.

Security architectures for WSNs rely upon cryptography operations as the basic method to achieve previously mentioned security requirements. Cryptographic schemes involve the adoption of one or more keys, used to encrypt and decrypt exchanged data; the main problem to face with when setting up a secure communication between nodes is *key agreement*, that is the way such keys are established at each node. There are two main well-known mechanisms to handle the problem of key agreement: Symmetric Key Cryptography (SKC) and Public Key Cryptography (PKC), the former adopting a unique secret shared key for both encrypting and decrypting messages, and the latter employing a couple of keys for each node, one public and the other private, resulting in an improvement of the security level of the system.

Several implementations of Symmetric Key Cryptography algorithms have been proposed in literature (i.e. Skipjack, DES, 3DES, AES, RC5 and RC6), as they require in general a reduced amount of computational resources and thus turn out to be well suited for realization on sensor devices. The feasibility of the software implementations of such algorithms was evaluated [7]-[9], concluding that the most effective algorithms such as RC4 and Skipjack have an overhead (energy, bandwidth, latency, and memory consumption) less than 10%; other algorithms such as AES impose a higher penalty over the resources of the node (around 20%), but they are still suitable for practical use.

Nowadays some implementations of complete secure protocols based on symmetric schemes are available, as TinySec [10], MiniSec [11], ZigBee [12] and SNEP [13]. TinySec, the first fully-implemented link layer security architecture, tightly coupled with the Berkeley TinyOS radio stack, achieves low energy consumption and memory usage, but it also sacrifices the level of security, not providing protection against replay attacks and employing a single network-wide key, such that every malicious node in the network can masquerade as any other node. ZigBee provides a higher level of security than TinySec since it is not restricted to a network-wide key and it protects against replay attacks, but it is an expensive protocol due to high communication overhead, high energy consumption by the radio and large memory utilization. MiniSec provides low energy consumption like TinySec, and a high level of security like ZigBee, while requiring less packet overhead, and has been showed to outperform other comparable systems under most real-world scenarios [11].

Even if symmetric schemes are very attractive for their energy and memory efficiency, they present a major drawback: key distribution and management are a fundamental concern, as they produce a heavy traffic in the network and often require complex and not scalable architectures. Actually, there has been a substantial amount of research on key distribution schemes [14]-[17], and two main solutions have been investigated: key pre-distribution, which involves assigning keys to a set of nodes before deployment according to deterministic or stochastic algorithms, and hierarchical schemes, relying upon a trusted controller for key assignment and exchange between nodes.

A further weakness of symmetric cryptography consists in that it only fulfills confidentiality requirements, while not considering other security issues such as authentication and integrity.

An important security requirement which arises within the sensor network domain is the *broadcast authentication*, that is the capacity of a sender to broadcast messages to multiple nodes in an authenticated way. In the two-party communication case, data authentication can be achieved through a purely symmetric mechanism making use of a *Message Authentication Code* (MAC), computed by the sender over the payload and appended to the message, in such a way that the packet is considered valid upon reception if the MAC recomputed by the receiver matches with the received one. This kind of solution is insecure in broadcast communication scenarios. In fact, anyone of the receivers knows the MAC key and could impersonate the sender. Instead, asymmetric schemes are the natural way for providing broadcast authentication.

Despite that, Perrig et al. [13] propose a key-chain distribution system for their μTESLA secure broadcast protocol, part of the SPINS system. The basic idea of the μTESLA system is that it constructs authenticated broadcast messages from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains. One of the limitations of μTESLA is that some initial information must be unicasted to each sensor node before authentication of broadcast messages can begin. To

face with these constraints enhancements to the μTESLA system have been proposed [18], [19]. However all of these schemes use symmetric key techniques with an elaborate design to add asymmetric properties to them and require loose time synchronization between nodes.

In summary, it can be stated that, like other main security requirements, broadcast authentication can be naturally achieved through asymmetric schemes. Such schemes do not need time synchronization and allow the introduction of digital signatures, by means of which a message can be quite easily associated with an entity, thus enabling authentication features.

The use of asymmetric schemes in sensor networks has been usually considered as "nearly impossible" because they are power consuming and require a large amount of computational and storage resources. However, as previously said, such schemes are very attractive, because they can ensure a higher degree of security while guaranteeing a greater flexibility and manageability than symmetric ones: thanks to them, any two sensors can establish a secure channel between themselves to distribute keys; moreover, as nodes do not share the same common key for encrypting/decrypting messages, the "capture" of some sensor devices will not affect the security of others.

Rivest-Shamir-Adelman (RSA) algorithm [20] and Elliptic Curve Cryptography (ECC) [21] are amongst the most well known public key algorithms used in security systems, the latter being an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Many papers and articles discussed the efficiency of each of these protocols, and showed that ECC is more efficient than RSA in terms of memory requirements because it requires much lower key size than RSA to achieve the same security level: it has been proved that ECC with 160-bit keys provides the currently accepted security level, and is equivalent in strength to RSA with 1024-bit keys (RSA-1024) [22].

At present some studies have demonstrated that with careful design, the Elliptic Curve Diffie-Hellman (ECDH) key agreement technique [21], based on Elliptic Curve Cryptography, can be deployed on even the most constrained of the current sensor network devices [23]-[26]. Moreover, Elliptic Curve Digital Signature Algorithm (ECDSA) [21], a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups, can be used for signature generation and verification.

In order to overcome the drawbacks of both PKC and SKC schemes, a hybrid approach could be adopted, by combining the higher security level accomplished by the first ones with the efficiency of the latter ones in terms of required resources. AL-Rousan et al. in [27] proposed a security system relying upon a symmetric key function for ensuring secure communication between in-network nodes, and a public key function for providing a secure data delivery between source nodes and the sink; the proposed scheme suits well to data-centric networks, in which only a subset of the fields in the exchanged packets is needed for aggregation at intermediate nodes, while the whole packet has to be seen only by the sink and the source. It suggests that a symmetric key algorithm should be used by the intermediate nodes to encrypt/decrypt the aggregation data portion (a common secret key shared by all nodes is used for this purpose), while the required data portion is encrypted/decrypted using a public key algorithm.

A hybrid approach, being slightly different by this one, can be considered for a tiered network composed of one master and many motes, each communicating only with the master: a public key function could be used to ensure authentication of the master and also to establish secret symmetric keys between the master and each of the motes, in order to ensure a higher level of security, while limiting as much as possible the cryptographic computational load.

III. Tenet Overview

The Tenet system [1], [2] is an architecture for tiered sensor networks which provides a high-level programming abstraction and allows applications to dynamically task and re-task the sensor network. The Tenet architecture is motivated by the observation that future large-scale sensor network deployments will be tiered [28], [29]: in a tiered architecture, nodes form a hierarchy in which each of them performs a specific set of tasks at a given level on behalf of a subset of nodes at the level below. This way they realize a functional decomposition which can reflect physical or logical differences among nodes [30]. Tiered architectures are scalable and cost-effective, as they allocate resources where they can be most efficiently utilized; moreover, organizing a network in tiers can increase network lifetime by partitioning different functions among specifically designed hardware platforms.

The Tenet system splits a sensor network into two tiers (Figure 1) : in the lower tier we can find simple sensor nodes, called "motes", which merely perform local processing on sensed data, while in the upper tier we find the "masters", rather unconstrained nodes performing multi-node data fusion and complex application logic, often provided with a consistent source of energy.

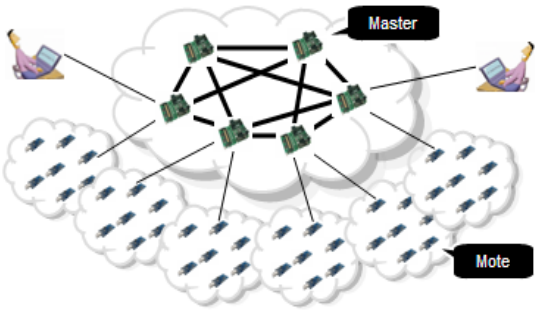

Figure 1. The Tenet architecture

The Tenet project's guiding architectural principle asserts that multi-node data fusion functionality and complex application logic should be implemented only on the masters, while allowing motes to process locally-generated sensor data. All communication to the mote tier consists of *tasks*, and all communication from the mote tier consists of task responses (such as sensor data) destined for a master and delivered to the application program (Figure 2). The master node can then fuse the results, re-task motes or trigger other sensing functionalities.

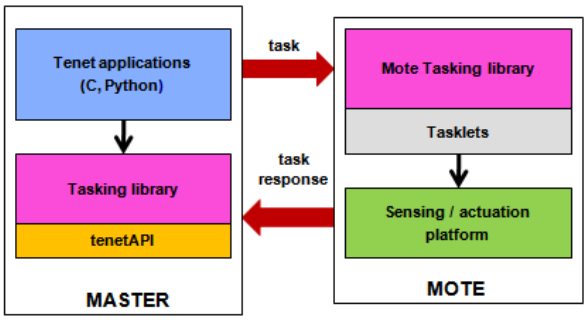

Figure 2. The Tenet programming model

Applications specify a task as a linear data flow program consisting of a sequence of *tasklet* implementing basic functionalities as timers, sampling, data compression, thresholding, statistical operations, and other forms of simple signal processing. For example, to construct a task that samples the temperature sensor every minute and sends the samples to its master, an application should construct the following task:

```
periodic(1 min) -> sample(TEMPERATURE)
-> Send()
```

In Tenet one or more applications run concurrently on the less constrained master tier, where programmers can use familiar programming interfaces (compiled, interpreted, visual ones) and different programming paradigms (functional, declarative, procedural ones), simplifying application development. At the same time, the mote tier networking functionality is generic, since Tenet's networking subsystem merely needs to robustly disseminate task descriptions to the motes and reliably return results to masters: this enables significant code reuse across applications and energy-efficient operations.
The Tenet task library was implemented on top of TinyOS, in order to get advantage of the robustness and availability of its drivers. Because of this choice it is impossible to dynamically load software libraries at runtime; thus, all tasklets an application might require must be compiled into a single binary.

Tenet is equipped with a networking sub-system which provides task dissemination, routing from motes to the master and end-to-end reliable transport. Any mote must be able to return a response to the tasking master: Tenet uses a novel *tiered routing* mechanism, where a mote's response is first routed to its nearest master, and is then routed on the master tier using an IP overlay. The routing system also enables point-to-point routing between masters and motes, necessary for example if a master has to adaptively re-task an individual mote or if a master has to directly send the task description to a specific mote instead of using Tenet task dissemination mechanism for efficiency purposes.

Tenet supports three types of delivery mechanisms, which applications can select by using the corresponding tasklet in their task description: a best effort transport, useful for loss-tolerant periodic low rate applications, a transactional reliable transport for events, and a stream transport for high-data rate applications, all of which using a limited number of hop-by-hop retransmissions to counter the high wireless packet loss rates encountered in practice.

IV. Proposal

As previously seen, security issues are a central concern for sensor networks, as they are often adopted in critical applications despite having many characteristics that make them very vulnerable to malicious attacks. Because of their resource constraints, it is very difficult to implement strong security algorithms on sensor platforms and there is still much work to do to address this matter. However, if we consider a tiered system such as Tenet, whose master layer nodes are supposed to have relatively more plentiful resources, we can assume that the most complex and power consuming operations are placed on such nodes: this way it is possible for example to perform some complex cryptographic algorithms exploiting the different capabilities of network components.

Hence, our proposal is the enhancement of the Tenet architecture by means of the introduction of a cryptosystem, in order to achieve some security requirements in a tiered network. As for now we will not cover aspects such as data fusion security, secure localization, secure time synchronization, secure routing and transport but we will keep such points as future works. Instead, our proposal aims at ensuring the following security properties:

- achieve end-to-end encryption, integrity and freshness of response packets sent by motes to the master;
- implement a mechanism for key exchanging (and storing) between the master and motes in such a way that different pairs of keys are kept between each mote and the master;
- achieve broadcast authentication of messages sent by a master to the motes;

As for the first point, we have adopted a symmetric scheme in order to efficiently ensure confidentiality, integrity and freshness of response packets sent by motes to the master: at this aim we have integrated the MiniSec architecture [11] with the Tenet system. As for the key exchanging and broadcast authentication protocols, they have been implemented by exploiting the TinyECC library [32], a publicly available software package for ECC operations including some optimization features which can be enabled/disabled through opportune software switches.

The key exchanging protocol is naturally achieved via Tenet tasking system itself, while as for broadcast authentication the only constraint is that each mote has to be preloaded with the public key of the base station. This is slightly acceptable since a Public Key Infrastructure for sensor networks still does not exist at the moment.

Current implementation of the cryptosystem has been realized by taking into account a single master Tenet architecture: we made no assumptions on master-to-master communication, but we have kept this point as a future work. In the following sections, we firstly give a brief overview of the adopted software packages (MiniSec and TinyECC) and then illustrate the design principles of our security scheme. We have implemented and tested the proposed architecture on TelosB motes and PC-class devices with Tenet-t1 running on top of TinyOS 1.x [31]. However a more complete evaluation of security features of our cryptosystem will be addressed in future works as well the porting of our code to Tenet-t2 running on top of TinyOS 2.x.

V. Security Enhanced Tenet Architecture

The design of the cryptosystem for the Tenet architecture focuses on exploiting low level security primitives provided by publicly available software packages. In this section, we first give an overview of the adopted tools, and finally illustrate the design as well as some implementation details of our cryptosystem and its integration with the Tenet architecture.

A. Adopted Technologies

MiniSec is a secure network layer that provides a high security level in terms of data confidentiality, integrity and freshness, while keeping low energy consumption. Minisec's source code is publicly available for Telos motes, but can be easily ported to other platforms. It has two operating modes, one tailored for single-source communication (unicast communication), and another tailored for multi-source broadcast communication.

Both schemes employ OCB or Offset CodeBlock [33] as encryption mode, which is especially well-suited for the stringent energy constraints of sensor nodes and is able to provide secrecy and authenticity in one pass of the block cipher. Data authentication is achieved by the sender by computing a Message Authentication Code (MAC) over the payload and appending that to the message, with the receiver having to recompute it and verify the matching with the received one. Also, MiniSec provides a mechanism to guarantee a "weak" level of freshness, based on the use of a counter as a nonce, by which a receiver can determine a partial ordering over received messages without a local reference time point.

Authors rewrote part of the TinyOS network stack, specifically the Active Message layer, responsible for managing the communication over the radio channel, in such a way all outgoing messages are encrypted, while all received packets are decrypted: this is done by appropriately modifying the GenericComm and AMStandard TinyOS core modules.

MiniSec uses 80-bit symmetric keys, considered to be secure until 2012. When 80-bit keys become insecure, it will be possible to use 128-bit AES keys [34], secure for the next 20 years. Minisec's packet format is based on the current TinyOS packet header for Telos mote's CC2420 radio, with the addition of a source address and a counter fields, resulting in an overall 3-byte overhead. In this way, MiniSec achieves the lowest communication overhead among its major counterparts (i.e. TinySec), with respect to a standard TinyOS network stack.

TinyECC is a configurable library for ECC operations in wireless sensor networks. Its primary objective is to provide a ready-to-use, publicly available software package for ECC-based PKC operations that can be flexibly configured and integrated into sensor network applications. TinyECC includes all the well-known ECC schemes, such as ECDH key agreement scheme and ECDSA digital signature scheme. It also includes a public key encryption scheme (ECIES) and some optimization features for ECC operations, which can be enabled/disabled by developers by means of apposite software switches. TinyECC has been tested on MICAz, TelosB, Tmote Sky, and Imote2 platforms running TinyOS. By default, TinyECC includes all 128-bit, 160-bit and 192-bit ECC parameters recommended by SECG (Standards for Efficient Cryptography Group) [21].

B. Design Overview

Figure 3 shows the security enhanced Tenet architecture, having been realized from the current Tenet prototype. Red dashed lines indicate new modules added to the system as well as extensions of existing ones with new components.

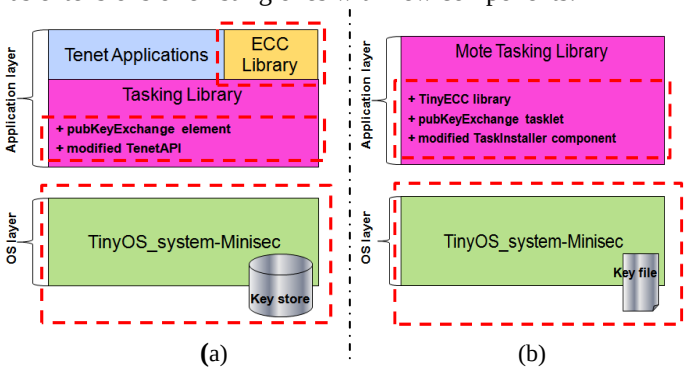

Figure 3. Modified Tenet stack on (a) master side and (b) mote side

The Tenet system can be considered as composed of 2 main software layers: an *application layer* and a *OS layer*, the latter being implemented by TinyOS [31], the most commonly used free and open source Operating System for wireless sensor networks; in order to enhance Tenet with security capabilities, we have integrated into this structure the libraries described above.

Let us first consider the master side (Figure 3.a), where we have modified its application layer structure by introducing an *ECC Library* based on the TinyECC distribution, and by adding the *pubKeyExchange element* to the *Tasking Library*, in order to let the Tenet system correctly interpret a task containing the pubKeyExchange tasklet. We have also modified the *TenetAPI* in order to implement the digital signature of task messages sent by master to motes.

The OS layer has been modified by integrating TinyOS with Minisec, responsible for cryptographic operations and management of the shared keys between the master and each of the motes.

On the mote side (Figure 3.b) we have improved the *Mote Tasking Library* by defining and implementing the *pubKeyExchange* tasklet, aimed to carry out security operations according to the ECDH key agreement technique.

Motes OS layer has also been modified in order to opportunely integrate the Minisec system.

C. Implementation details

Let us now describe more in detail the implementation of key agreement, broadcast authentication and end-to-end encryption operations in our system.

As previously said, as for **key establishment** we have implemented the ECDH key agreement protocol by exploiting the Tenet tasking system and TinyECC primitives. In a key establishment scenario the master sends to each mote the following task:

```
pubKeyExchange(PPx, PPy) -> Send()
```

where *pubKeyExchange* is a new tasklet added to the Mote Tasking Library, that aims to perform ECC security operations according to the ECDH key agreement technique, and PPx and PPy are the coordinates of master's Public Point.

As applications running on the master are written in the C language, we ported TinyECC code from nesC to C, thus constructing the *ECC Library* exploited by the master in order to perform ECC security operations. Also, we added the *pubKeyExchange element* to the Tasking Library on master side, in order to let the Tenet system correctly interpret a task containing the pubKeyExchange tasklet.

The ECDH protocol has been implemented according to the following steps, illustrated in Figure 4:

1. the master runs the ECDH application and initializes the Elliptic Curve;
2. the master calculates its Public Point on that curve and
3. sends the previously mentioned task to the mote with the two coordinates (PPx, PPy) of its Public Point;
4. the mote initializes the Elliptic Curve and
5. calculates its public point on that curve;
6. the mote calculates the shared secret, that is its own private key shared with the master, and stores it in the MiniSec *keyfile*;
7. the mote sends the task response with the two coordinates of its Public Point
8. finally, the master calculates the shared secret and stores it in the MiniSec *keyfile*. The master keeps a list of as many keyfiles as the number of motes.

The procedure is iterated for all motes, in such a way that each of them shares a different private key with the master.

The calculated shared secret is a 160-bit key, however only the first 80 bits will be stored in the keyfiles, composing the cryptographic symmetric key employed by the Skipjack cipher within MiniSec.

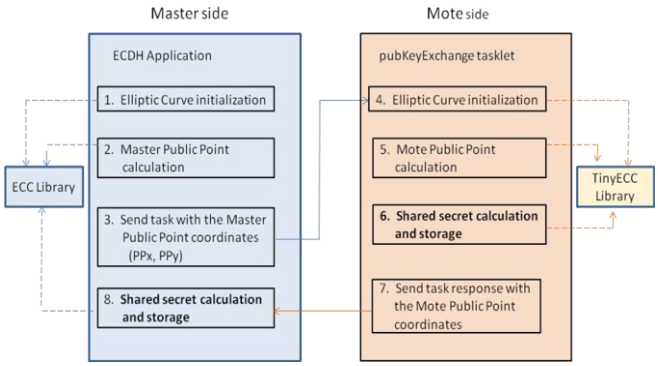

Figure 4. ECDH key agreement protocol using the Tenet tasking system

As for **broadcast authentication**, we assumed that broadcast tasking messages from master to motes must be authenticated in such a way each mote can verify the identity of the master node. Hence, we have implemented the ECDSA scheme by using again the primitives provided by TinyECC. The only constraint is that during the initialization phase of the system the master should generate a key pair (private key – public key) and store its private key in the ECC Library.

On the other side, each mote should be preloaded with the public key of the master, opportunely stored in the TinyECC Library. That assumption can be accepted since there is not yet a Public Key Infrastructure for public key distribution in sensor networks. On master side, tasking messages are signed with the master private key in the TenetAPI module and sent to motes together with the signature. On mote side the signature is verified in the *TaskInstaller* component with the master public key. The high modularity of the Tenet system allowed us to easily add security operations into the above mentioned opportune elements.

Finally, as for **confidentiality**, **integrity** and **freshness** of task response messages from motes to the master, we have opportunely integrated the MiniSec security layer into the Tenet system. As previously mentioned, MiniSec's authors simply rewrote the ActiveMessage layer of the TinyOS network stack for encrypting all outgoing messages and decrypting all received ones. Since we are just interested in securing task response messages, on mote side we integrated the MiniSec *AMStandard* module and modified it in such a way it only does encryption of outgoing task response messages which are identified with a specific *tag*; on master side we added MiniSec decrypting operation into the *AMFiltered* component running on the base station in order that it just decrypts incoming task response messages identified with the above mentioned specific tag. Obviously, those operations are performed by using previous exchanged private keys between the master and each mote.

VI. Conclusion and Future Works

In this paper, we have proposed the design of a hybrid cryptosystem aimed to secure the Tenet architecture. We have combined symmetric and asymmetric cryptographic schemes in order to achieve key exchange mechanisms (through the definition of a specific tasklet added to the Tenet Tasking Library), end-to-end encryption, integrity and freshness of response packets sent from motes to the master, and broadcast authentication of tasking messages coming from the master to motes. These goals have been reached by opportunely integrating the TinyEcc library and the Minisec security layer with the Tenet architecture.

We have implemented and tested our schemes for Telos motes running Tenet-t1 on top of TinyOS 1.1.x. Future works will be devoted to port our code to TinyOS 2.x in order to be compliant with Tenet-t2 release as well as to port it to other sensor platforms.

Actually, the development of a cryptosystem based on the Tenet architecture is not an end in itself: our main goal is to set up different security protocols and architectures based on well-known or novel solutions, in order to develop a general design methodology for wireless sensor networks having strict security requirements. At this aim, we plan to try out different security schemes on more complete testbeds in order to be able to evaluate such solutions in terms of the tradeoff between the achieved security level and the resulting performances.

Acknowledgment

We thank Ramesh Govindan for having inspired and followed out this work. We also thank all members of the Embedded Networks Laboratory of the University of Southern California, above all Omprakash Gnawali,

Jeongyeup Paek and Marcos Vieira for their advice and collaboration.

References

[1] J. Paek, B. Greenstein, O. Gnawali, K.-Y. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, E. Kohler, "The Tenet Architecture for Tiered Sensor Networks", *ACM Transactions on Sensor Networks (TOSN)*, VI (4), pp. 1-44, 2010.

[2] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, and E. Kohler, " The TENET Architecture for Tiered Sensor Networks", *In Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, pp.153-166 , 2006

[3] J. Hicks, J. Paek, S. Coe, R. Govindan, and D. Estrin, "An Easily Deployable Wireless Imaging System", *In Proceedings of the Workshop on Applications, Systems, and Algorithms for Image Sensing (ImageSense)*, 2008.

[4] J. Paek, O. G. K.-Y. Jang, D. Nishimura, R. Govindan, J. Caffrey, M. Wahbeh, and S. Masri, "A Programmable Wireless Sensing System for Structural Monitoring", *In Proceedings of the 4th World Conference on Structural Control and Monitoring (4WCSCM)*, 2006.

[5] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor Network Security: A Survey", *IEEE Communications Surveys & Tutorials*, XI (2), pp. 52-73, 2009

[6] D. Dolev and A.C. Yao, "On the security of public key protocols", *In Proceedings of the IEEE 22nd Annual Symposium on Foundations of Computer Science*, pp. 350-357, 1981.

[7] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu."Analyzing and Modeling Encryption Overhead for Sensor Network Nodes". *In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA'03)*, San Diego, USA, September 2003

[8] K. Jun Choi, J.-I. Song. "Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Networks". *In Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT'06)*, Phoenix Park, Korea, February 2006.

[9] Y. W. Law, J. Doumen, P. Hartel. "Survey and Benchmark of Block Ciphers for Wireless Sensor Networks". *ACM Transactions on Sensor Networks*, II (1), pp. 65-93, February 2006.

[10] C. Karlof, N. Sastry, and D. Wagner, "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks", *In Proceedings of the 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, November 2004.

[11] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: A Secure Sensor Network Communication Architecture", *In Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007)*, April 2007.

[12] ZigBee Alliance, "Zigbee specification", *Technical Report Document 053474r06, Version 1.0*, ZigBee Alliance, June 2005.

[13] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks", *ACM Wireless Networking*, VIII (5), pp. 521–534, 2002.

[14] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks", *IEEE Symposium on Research in Security and Privacy*, pp. 197–213, 2003.

[15] W. Du, J. Deng, Y. S. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks", *In Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, pp. 42–51, 2003.

[16] W. Zhang, S.Zhu, and G. Cao, "Predistribution and local collaboration-based group rekeying for wireless sensor networks", *ELSEVIER Ad Hoc Networks* VII, pp. 1229–1242 , 2009.

[17] H. T. T. Nguyen, M. Guizani, M. Jo, and E. Huh, "An Efficient Signal-Range-Based Probabilistic Key Predistribution Scheme in a Wireless Sensor Network", *IEEE Transactions On Vehicular Technology*, LVIII (5), pp. 2482 - 2497 , 2009.

[18] D. Liu and P. Ning, "Multi-level mTESLA: Broadcast authentication for distributed sensor networks", *ACM Transactions in Embedded Computing Systems (TECS)*, III (4), pp.800–836, 2004.

[19] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks", *In Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*, pp. 118-129, 2005.

[20] R.L. Rivest, A. Shamir, and L.A. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM* XXI (2), pp. 120–126, 1998.

[21] Certicom Research, "Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography", Version 1.0, September 20, 2000.

[22] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz "Comparing elliptic curve cryptography and RSA on 8-bit CPUs", *In Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, Cambridge, MA, pp.119-132, 2004.

[23] J. Lopez, "Unleashing Public-Key Cryptography in Wireless Sensor Networks", *Journal of Computer Security*, XIV (5), pp 469-482, 2006.

[24] G. Gaubatz, J. Kaps, and B. Sunar, "Public keys cryptography in sensor networks" , *In Proceedings of the 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS)*, 2004.

[25] E.O. Blaß and M. Zitterbart, "Towards acceptable public-key encryption in sensor networks", *In Proceedings of the 2nd ACM International Workshop on Ubiquitous Computing*, pp.88–93, INSTICC Press, Miami, USA, May 2005.

[26] R. Watro, D. Kong S. Cuti, C. Gardiner, C. Lynn and P. Kruus, "TinyPK: Securing Sensor Networks with Public Key Technology", *In Proceedings of the 2nd ACM Workshop on Security of ad hoc and Sensor Networks (SASN 2004)*, Washington, DC, USA, October 25, 2004.

[27] M. AL-Rousan, A. Rjoub, and Ahmad Baset, "A Low-Energy Security Algorithm for Exchanging Information in Wireless Sensor Networks", *Journal of Information Assurance and Security IV*, pp. 48-59, 2009.

[28] A. Arora et al, "ExScal: Elements of an extreme scale wireless sensor network", *In Proceedings of the 11th IEEE International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA '05)*, August 2005.

[29] R. Guy, B. Greenstein, J. Hicks, R. Kapur, N. Ramanathan, T. Schoellhammer, T. Stathopoulos, K. Weeks, K. Chang, L. Girod, and D. Estrin, "Experiences with the Extensible Sensing System ESS", *Technical Report 61*, CENS, Mar. 29 2006.

[30] Mohammad Ilyas and Imad Mahgoub. *Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems*. CRC Press 2005. Print ISBN: 978-0-8493-1968-6

[31] TinyOS Project, URL: http://www.tinyos.net.

[32] A. Liu, P. Kampanakis, and P. Ning. "TinyECC: Elliptic curve cryptography for sensor networks", *In Proceedings of the 7th International Conference on Information Processing in Sensor Networks, IPSN 2008*, St. Louis, Missouri, USA, April 22-24, 2008.

[33] P. Rogaway, M. Bellare, J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption", *In Proceedings of the ACM Transactions on Information and System Security (TISSEC)*, VI (3), pp.365-403, 2003.

[34] http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html.

Author Biographies

Alessandra De Benedictis is currently a Ph.D. student in Computer and Control Engineering at the University of Naples Federico II. She received a B.S. degree and an M.S. degree in Computer Engineering, both summa cum laude, from the University of Naples Federico II in March 2009. Her research activities include security in Wireless Sensor Networks, Embedded System Design and performance evaluation.

Andrea Gaglione received a B.S. degree and an M.S. degree in Computer Engineering, both summa cum laude, from the Second University of Naples in 2004 and 2006, respectively. He got a Ph.D. in Computer and Control Engineering from the University of Naples Federico II in 2009 and his research activities include Sensor Networks, Event Recognition, and Critical Infrastructure Protection.

Nicola Mazzocca is a full professor of High-Performance and Reliable Computing at the Computer and System Engineering Department of the University of Naples Federico II, Italy. He owns an MS Degree in Electronic Engineering and a Ph.D. in Computer Engineering, both from the University of Naples Federico II. His research activities include methodologies and tools for design/analysis of distributed systems; secure and real-time systems and dedicated parallel architectures.