IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Key Management Systems for Sensor Networks in the Context of the Internet of Things

DHIVYA. R,
PHD PART TIME,
DEPARTMENT OF COMPUTER SCIENCE,
GOBI ARTS & SCIENCE COLLEGE ,
KARATTATIPALAYAM,
GOBICHETTIPALAYM,
ERODE(DT).

DR. B. SRINIVASAN,
ASSOCIATE PROFESSOR,
DEPARTMENT OF COMPUTER SCIENCE,
GOBI ARTS & SCIENCE COLLEGE ,
KARATTATIPALAYAM,
GOBICHETTIPALAYM,
ERODE(DT).

Abstract

If a Wireless Sensor Network (WSN) is to be completely integrated into the Internet as partof the Internet of Things (IoT), it is necessary to consider various security challenges, suchas the creation of a secure channel between an Internet host and a sensor node. In order to create such channel, it is necessary to provide key management mechanisms that allow two remote devices to negotiate certain security credentials (e.g. secret keys) that will be used to protect the information flow. In this paper we will analyse not only the applicability of existingmechanisms such as Public Key Cryptography and pre-shared keys for sensor nodes in the IoT context, but also the applicability of those link-layer oriented key management systems (KMS) whose original purpose is to provide shared keys for sensor nodes belonging to the same WSN.

Keywords: Wireless Sensor Networks, Internet of Things, Security, Key Management

1. Introduction

The vision of the Internet of Things is, as of 2010, in an embryonary state. Most of the elements of our world, our things, do not have the digital intelligence that enables them to be aware of the existence of a virtual cyberworld, where they could be able to collaborate with literally billions of entities: other things, humans, computational processes, our own physical environment, and so on. But this vision is being pursued with tenacity, and the technologies that will enable it (e.g. RFID, Wireless Sensor Networks, Machine-to-Machine interactions) areslowly but inexorably becoming part of our daily lives.

Whether we achieve the complete vision of the Internet of Things or only a part of it dependson multiple factors. One of those factors is security. The Internet of Things must comply with multiple security requirements (e.g. privacy, robustness), in order to provide a strong foundationthat will be resilient against those who will try to take advantage of its existence. Not only the integration of all the elements that will compose the Internet of Things must be secure, but alsothose elements must be inherently secure by themselves. As a result, there are multiple securitychallenges that must be taken into account.

Focusing on one of the technologies of the Internet of Things, Wireless Sensor Networks, a particularly important challenge is the creation of an end-to-end secure channel between remote

entities. Sensor networks can allow things to know the state of their surroundings and commu- nicate with other entities. Therefore, it is necessary to allow the elements of a sensor network, the sensor nodes, to connect other entities through the Internet. However, the issues surrounding the protection of the information flow are not trivial. Sensor nodes are usually constrained de- vices with limited computational power, and the key management mechanisms used to negotiatea session key with other entities might be too heavy for them.

Therefore, it is the purpose of this paper to analyse how existing key management systems that are frequently used in actual Internet scenarios could be applied in Internet-enabled Sen- sor Networks. However, in our study we will go one step further, and also evaluate those key management mechanisms that are focused on the negotiation of link-layer keys between neigh- bouring nodes. Obviously, such mechanisms were not designed with an Internet scenario in mind. Nevertheless, the existing corpus of research on this particular field is large enough to warrant an investigation on their applicability.

The structure of this paper is as follows. Section 2 provides an introduction to the Internet ofThings and the role of Wireless Sensor Networks in this new vision. Section 3 describes the mostrelevant security challenges of Internet-enabled Sensor Networks, focusing on the problem of es-tablishing a secure channel with remote entities. Moreover, we also state the assumptions and methodology of our analysis. In section 4, we analyse the applicability of Public Key Cryptog- raphy and the different pre-shared keys strategies, while in section 5 we focus on the analysis ofexisting link-layer oriented key management systems. Finally, we present our main conclusionsand future work in section 6.

2. The Role of WSN in the Internet of Things

An Overview of the Internet of Things

As of 2010, The Internet of Things, or IoT, is a vision of the future. Its definition is still fuzzy [1, 2], but all the people that share such vision also share the same belief: it is possible to create a world-wide network of interconnected objects, or things, that will have an active role inthe Future Internet. Such things will probably be readable, recognizable, locatable, addressable, and/or controllable via the Internet [3]. Moreover, these things do not need to be the electronic objects we encounter everyday, such as computers, but also other products like cars, books, or even food. Actually, within this vision our very households and the buildings that surround us could be considered things, filled with other things.

These things, that belong to the real/physical world, will make use of different technologies (e.g. embedded systems) to create a digital/virtual space, where they will interact with each otherand other entities (e.g. software processes, human beings) through well-defined interfaces. As a consequence, things are expected to become active participants in business, information and social processes. Observe that this digital space will be tightly related to the real world: not onlythe things will be able to maintain their real identity in the virtual world, but also they will accessto information from the real world (e.g. their physical location, the state of their environment) and interact with real-world entities.

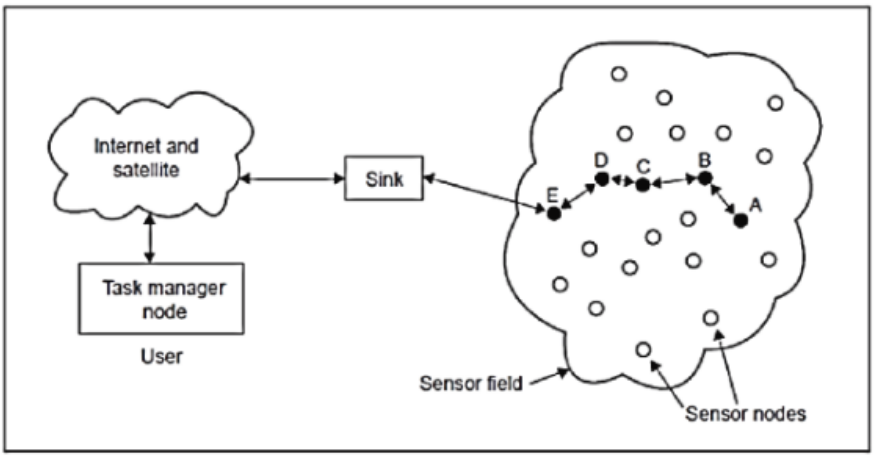

Fig1. Wireless Sensor Network

As for the real improvements that the IoT can bring to humanity, this vision creates new opportunities in various application domains. For example, the industrial sector can make use of the IoT services to improve the financial or commercial transactions between entities. Also, our environment can be effectively protected and monitored, and new services can be createdto foster the development and inclusion of societies, cities and people. Particular examples of

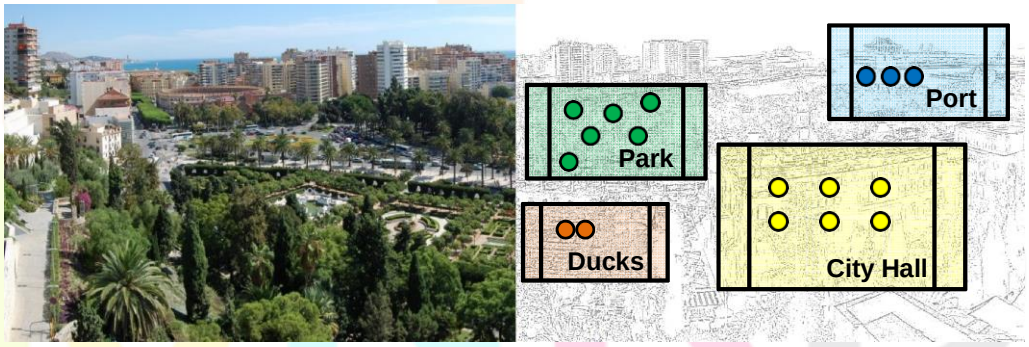

Figure 1: Real World, Virtual World, and WSN

specific services within these application domains are human activity monitoring (e.g. healthcareand elderly care), infrastructure integrity management (e.g. critical infrastructure supervision), and energy efficiency (e.g. smart energy metering, efficient consumption by ehicles) [2].

There are a wide range of technologies that will be involved in building the IoT. The enhance- ment of the communication networks infrastructure (e.g. through ultra-wideband networks, 3G and 4G networks) will be essential, as well as the adoption of IPv6 in order to provide a unique IP address to each thing involved in the network. The technologies that allow the location and identification of physical objects (e.g. RFID) will also be basic in this context. There are also other technologies that will influence on the successful development of IoT applications, such ascomputer vision, biometric systems, robotics, and others. One of these technologies, Wireless Sensor Networks [4], is able to provide an autonomous and intelligent link between the virtual world and the physical world, and in fact it has been thoroughly studied. We will focus on WSNon this article, as it brings the IoT one step closer to reality.

Sensor Networks in a Globally Connected World

WSNs have evolved considerably in the last few years, turning from a promising research field into an efficient and profitable technology in certain fields (e.g. industrial environments [5]). The core element of a WSN is the sensor nodes, low-powered and resource-constrained devicesthat can 'feel' (gather physical data), 'think' (process data and make informed decisions), and 'talk' (communicate with other entities using a wireless channel). These sensor nodes (or simplynodes) collaborate with each other in a distributed manner towards the

same goal: to provide anyvirtual entity with information from its physical surroundings, such as temperature, humidity, light, or even radiation.
The role of WSN in the IoT is that of a virtual skin: by becoming part of those things (e.g. teddy bears, appliances, buildings, environments), it allows them to become aware of their sur- roundings and to share this information with other things in order to take informed decisions (cf. Fig. 1). For example, a teddy bear can detect if it is being hugged by its owner so as to play a lul- laby if (according to an Internet time service) it is time to go to bed. At a greater scale, a buildingcan be able to know about the state of its elements, such as classrooms, offices, laboratories, andrestrooms, and change itself accordingly (e.g. turn lights off at night in unoccupied rooms). Evenmore, an intelligent city can be able to aggregate the information coming from its infrastructures

(e.g. transportation, power grids, healthcare, water distribution systems), achieving a holistic point of view that will be crucial in any decision-making processes.
In fact, according to the academic community, WSN are expected to provide interconnectedsolutions for aircraft monitoring systems, intelligent buildings, healthcare environments, phar- maceutical products, processing industries, and environment monitoring [2]. Moreover, the in- tegration of WSN in the IoT is also a fact supported by several governments and international companies. Noteworthy examples are 'A Smarter Planet' [6], a strategy developed by IBM which considers sensors as fundamental pillars in intelligent water management systems and intelligent cities; and the CeNSE project by HP Labs, focused on the deployment of a worldwide sensor network in order to create a "central nervous system for the Earth" [7].
The technologies that will enable the integration of WSN with the IoT are being currently de- veloped and tested. For example, the 6LoWPAN standard, defined by IETF [8], allows the trans-mission of IPv6 packets through computationally restricted networks. Also, the ROLL routing standard aims to provide end-to-end routing solutions for industrial, connected home, building and urban WSN [9]. Moreover, there are ongoing standards that try to implement web services in the sensor nodes themselves, such as the Constrained Application Protocol or CoAP [10]. Note that it is also possible to link the data produced by the sensor nodes and retrieved by a front-end system (i.e. base station) with web services based on SOAP and REST [11], mes- saging mechanisms (such as emails and SMS) or social networks (e.g. Twitter) and blogs (e.g. Wordpress) [12].

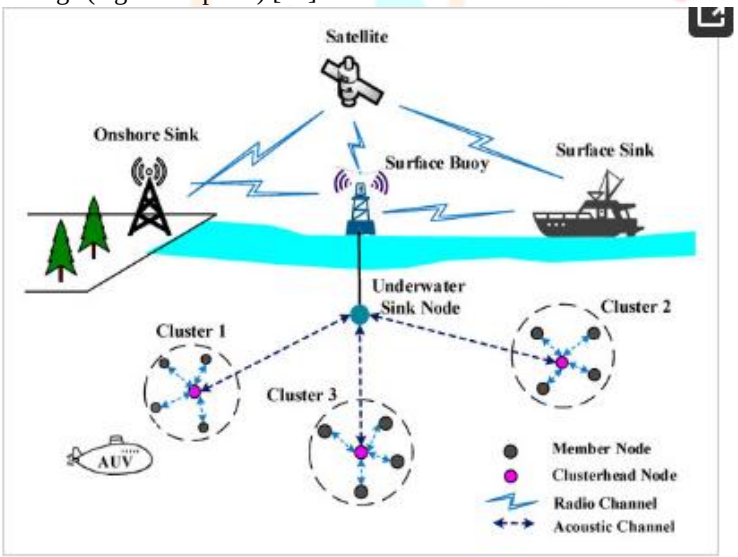

Figure 1. Underwater wireless sensor network (UWSN) architecture.

Precisely, one particular issue that must be considered when connecting a WSN to the Internet is the following: should sensor nodes provide their services directly (e.g. a thing that is equippedwith a sensor node provides a web service interface), or it is better to delegate this task to the base station? It has been mentioned before that a thing should be "locatable, addressable [...] via the Internet" [1], but this particular configuration might not be suitable for certain scenarios.There have been various studies that analyse this particular issue [13, 14]. Their conclusion is the following: in some specific scenarios (e.g. SCADA systems [15]) a sensor node does not need to provide its services directly, but there are other scenarios (e.g. First Responders) where a sensor node should be completely integrated into the Internet.

3. Security Issues of a Globally Connected WSN

In order for WSN to become an intrinsic part of the IoT, it is necessary to consider various challenges: from the adaptation of existing Internet standards to the creation of interoperable protocols and the development of supporting mechanisms for composable services [2]. Not surprisingly, one of these challenges is security, mainly because it is not possible to directly apply existing Internet-centric security mechanisms due to the intrinsic features of WSN (e.g. thecapabilities of the nodes, the bandwidth of the wireless channel). On the following paragraphs, we will highlight the most relevant security challenges that are related to the integration of WSNwithin the IoT. Note that these challenges are tightly related to WSN, but can also be applicable to other relevant technologies of the IoT (e.g. embedded systems, mobile phones, RFID).
Even if a WSN itself is protected with its own security mechanisms (e.g. using the link- layer security of IEEE 802.15.4), the public nature of the Internet will require of the existence of *secure communication protocols* for protecting the communications between two peers. As aforementioned, sensor nodes can make use of the 6LoWPAN protocol to interact with a IPv6 network (i.e. the Internet). Moreover, they are powerful enough to implement symmetric key cryptography standards such as AES-128 [16], and can be able to implement certain Random Number Generators [17]. However, mainly due to the power constraints and limited computa- tional capabilities of the nodes, at present there is no explicit support for the IPsec protocol suite in 6LoWPAN [18]. As a consequence, it is

necessary to study how other mechanisms could be used in order to create an end-to-end secure channel, such as TLS/SSL at the transport layer [19], or WS-Security at the application layer if SOAP-based web services are used [20]. Precisely, the implementation of *Secure Web Services* in constrained nodes is another issue that must be care-fully considered.

The creation of these secure channels is just one of the steps on the creation of a securely integrated WSN. Not all the services provided by a WSN will be public, thus we must develop *Authentication and Authorization* mechanisms in order to avoid unauthorized users (humans, machines) to access to the functionality of the WSN. This is not a trivial task, as the inherent distributed nature of the elements of a WSN must be considered. Moreover, we need to create suitable and scalable *Identification* mechanisms that can provide 'unique identifiers' and 'vir- tual identifiers' to all the different actors - WSN, sensors, users, and other devices. Finally, we must take into account the *survivability* problem of Internet-connected WSN. As with any other Internet server, a sensor node will surely be targeted by malicious entities trying to hinder the availability of its services. There have been some preliminary works on the development of In- trusion Detection Systems (IDS) for this particular environment [21], although more research is necessary if we want to reach our goal of a self-repairing, robust network.

Other important challenges on this particular field are the *integration of security mecha- nisms* [14] and *data privacy* [22]. Regarding integration, we must consider the security of the IoT from a global perspective. Even if different technologies such as WSN and RFID are se- cure by their own, their integration will surely generate new security requirements that must be fulfilled. Furthermore, it is necessary to analyse how the security mechanisms that protect one single technology will be able to coexist and interact with each other. As for privacy, recall the concept of WSN as a bridge between the real world and the virtual world. However, such bridge can be also abstracted as an omnipresent, intangible entity that behaves as an information high-way. This situation raises various questions: Who owns the data? How an user can be sure that his data is safe and will not be used without his consent? How personal data can be disclosed and used by authorized parties?

An Special Case: Secure Channels, Key Management

As we have already mentioned in the previous section, one of the challenges of connecting a WSN to the Internet is the creation of a secure channel that can adequately protect the informa- tion flow. This particular problem is simplified whenever the sensor nodes delegate all Internet connectivity to a base station (BS). This BS it is a more powerful device that usually behaves as an interface between the services provided by the sensor nodes and the users of the network. From the point of view of any Internet client (e.g. Things, Internet hosts), the real server will be the BS. Once the BS receives a query, it will forward it to a sensor node and wait for its reply. In case a sensor node needs to open a channel with an external server, it will simply forward the query to a BS, which will make the query on its behalf. Therefore, the BS will behave as any other web server, implementing any necessary security mechanisms. Also, the communi- cations between the sensor nodes and the BS will be protected using the WSN specific security mechanisms (e.g. shared keys between the nodes and the BS).

On the other hand, it has been mentioned (implicitly and explicitly) that some scenarios will require of a direct channel between an Internet host and a sensor node (cf. sections 2.1 and 2.2). For those scenarios, amongst other things, it is necessary to provide *key management mechanisms* that allow two remote devices to negotiate certain security credentials (e.g. secret keys) that will be used to protect the information flow. As in this context sensor nodes become full-fledged citizens of the Internet, we should examine at the existing security mechanisms and protocols used on the Internet that provide this functionality. For example, with TLS, two peers can negotiate a common key for ensuring secure end-to-end transit at the transport layer. Also, at the web services layer, WS-Security and its associated specifications can be used to set up a secure channel in the general Web services framework, where languages and protocols such as WSDL, UDDI, and SOAP are used.

In TLS, a client and a server negotiate a stateful connection by using a handshaking proce- dure. This procedure is as follows [19]. First, both peers exchange hello messages to agree on the ciphersuite (i.e. cryptographic algorithms) to be used, exchange random values, and check for session resumption. Later, they exchange certificates and cryptographic information to authen- ticate themselves and to provide session information, which will be used to generate the shared secret. Finally, the client and the server verify that the handshake has finished correctly without tampering by an attacker. Most of the ciphersuites require of Public Key Cryptography (PKC) in order to authenticate the server (and the client) and to derive the shared secret, although the nego-tiation can make use of a pre-shared key by using the TLS-PSK ciphersuite [23]. Note that TLS is designed to be used with reliable transport channels such as TCP, but a datagram-compatible variant is also available [24].

As for web services security, the WS-Trust specification provides a framework for requesting and issuing security tokens, allowing applications to construct trusted SOAP message exchanges. This functionality is used by WS-SecureConversation [25], allowing parties to exchange multiple messages in a secure fashion for the lifetime of a communications session. There are three different ways for establishing the security tokens: a) created by a trusted third party known as security token service (STS), b) created by one of the agents and propagated with a message, and

c) created by negotiation. In fact, the negotiation extensions allow both requestors and recipients to perform a set of exchanges prior to returning a security token.

Both TLS and the web services security protocols provide mechanisms for negotiating a shared secret between two peers. Most of these negotiation mechanisms are based either on Public Key Cryptography or on the existence of pre-shared master secret keys. By using PKC, two devices can exchange random values in a secure way (e.g. through PKC encryption) in order to generate a session key. As for pre-shared keys, these keys can be shared in advance amongst the communication parties, who will use them to create the session key. However, as PKC is supported but quite time consuming in constrained sensor nodes [26], and the use of a single pre-shared key might be unsuitable for certain IoT applications, it is necessary to analyse whether these key management mechanisms can be directly applied to Internet-enabled sensor nodes. It is important to point out that there are existing works that partially discuss this particular issue [27], and any reader interested in this area should also consider them.

Another question that might arise at this point is the following: In an Internet-enabled envi- ronment, can we reuse the key management systems (KMS) used to bootstrap a link-layer key between sensor nodes, using them to negotiate a session key between remote entities? Granted, there are major differences in the design of protocols that establish a session key between remote entities with those that try to provide link-layer credentials to physical neighbours. Moreover, these KMS usually implement specific optimizations (e.g. use the

physical location of the nodes)taking advantage of the properties of the WSN environment.
Nevertheless, the underlying goal of these protocols is, in fact, the same: to create a common key between peers. Moreover, this particular research field is quite mature, as suggested by the high number of published papers and surveys in the area. For example, a simple search in databases of abstracts and citations, such as Scopus[1] or Google Scholar[2], reveals that there are at least 400 scholarly articles written after 2002 on this particular area. As a consequence, it is worth analyzing whether these link-layer KMS protocols, which are designed with the limitationsof WSN in mind, could be used in an Internet environment to establish a direct secure channel between a sensor node and an Internet host.

Assumptions and Methodology

As our main goal is to analyse if Public Key Cryptography, pre-shared key strategies, and link-layer oriented Key Management Systems can be used to create secure channel between remote (constrained) entities, it is necessary to explain our assumptions and to state our method-ology. Regarding the assumptions about the network structure and its elements, they are the following:

Accessing the WSN. The routers that connect a WSN with the rest of the Internet must be able to convert any packets from IPv6 (Internet) to 6LoWPAN (WSN) and viceversa, as packets coming from both networks cannot be directly mapped. In this analysis we assume that such routers are in fact the base stations from the WSN, thus they can perform monitoring and management functions in certain situations.

Network structure. There are many different types of WSN, ranging from sensor net- works located on the ocean floor with specific transceivers (underwater WSN [28]) to sensor networks where the router is only available at certain moments in time (unattendedWSN [29]). In this analysis we will make use of the most "traditional" type of sensor network: a group of sensor nodes scattered in a certain area (e.g. a building) using IEEE
802.15.4 transceivers and constantly connected to the Internet through a set of routers /
base stations.

Sensor node features. There are different classes of sensor nodes, ranging from "Class I" nodes with roughly 1kB of RAM and 16kB of instruction memory (e.g. Arduino Duemi- lanove), to powerful "Class III" nodes with 256kB of RAM and 4MB of instruction mem-ory (e.g. Sun SPOT). In this analysis we will focus on the most common "Class II" node (e.g. MICAz, IRIS). This type of node has 4-16kB of RAM and 48-256kB of instruction memory, and are equipped with a 4Mhz-8Mhz microcontroller [30]. *Sensor node behaviour.* In most cases sensor nodes are assumed to behave as servers (server nodes), implementing web servers and providing services. However, in this anal- ysis we will also assume that sensor nodes can also behave as web clients (client nodes). Therefore, sensor nodes can be regularly contacted by external entities ('pull'), but also can send information to external servers at predefined intervals ('push'). In our analy- sis, whenever we mention "clients" and "servers" we will refer to both sensor nodes and Internet hosts, while "client nodes" and "server nodes" refer only to sensor nodes.

Service model. Although a sensor node can behave either as a client or as a server, in our analysis we will only consider the existence of the client/server model. That is, we will

not consider the existence of P2P applications where multiple sensors and Internet hosts behave as clients and servers. Besides, we will not consider the existence of trusted third parties that could assist on the establishment of a secure channel.

Regarding our methodology, we will follow a property-oriented analysis. Every key manage-ment system has some properties, which are discussed below, that are relevant for the creation ofa session key between remote entities. We will use these properties to compare the different keymanagement systems and their suitability for the IoT.

Distribution. This property is related to how the information that is needed for the ne- gotiation of a shared secret (e.g. public key certificates, pre-shared keys) is distributed. Distribution is considered as *offline* if all the information is stored beforehand within the peers. A distribution can also be considered as *online* if part of the information can be distributed during the negotiation process. An online distribution is desirable in the IoT,
as any client would be able to establish a secure channel with any server anytime, even ifthey did not meet before.

Authentication. This property defines whether the peers (*client* and/or *servers*) are au- thenticated during the negotiation process or not. This can be achieved, for example, usingof a uniquely shared key or a PKC signature. Note that in the negotiation process it is possible to authenticate a peer (e.g. if a pre-shared key is shared only with that peer) or a group (e.g. if a pre-shared key is shared with a group of peers). For a IoT context, server authentication is desirable, in order to assure clients that they are obtaining data from the right source.

Overhead. This property specifies both the communication and computational overhead of executing the negotiation process. This property will refer mainly to the overhead im- posed in sensor nodes, as i) they are much more resource constrained than other Internet hosts and ii) the energy consumed by sending and receiving information through the wire-less channel is quite high, and most sensor nodes will be battery-powered. As expected, the overhead in the IoT context must be as low as possible.

Resilience. This property indicates the ability to cope with stolen credentials. The re- silience of a key management system is low if an attacker that extracts information from one single device (e.g. Internet host, sensor node) is able to access all the secure infor- mation flows. In contrast, resilience is high if an attacker can only impersonate the devicethat was attacked. Unsurprisingly, a high resilience is desirable for IoT devices.

Scalability. This property is tightly linked with the amount of information that must be stored within a device in order to negotiate a secure channel with as much entities as possible. Considering that the potential Internet servers that can be accessed by a certain client

(and viceversa) are huge, a key management system is not scalable if the amount ofpreloaded information that must be stored inside a device grows linearly with the number of potential client/servers. On the other hand, a key management system is scalable if the information required to contact any potential peer does not impose an overhead to the device. Again, it is desirable to have a high scalability.

Extensibility. A key management system is not extensible if the number of peers that canbe securely contacted through a negotiation process is limited to a certain number. While

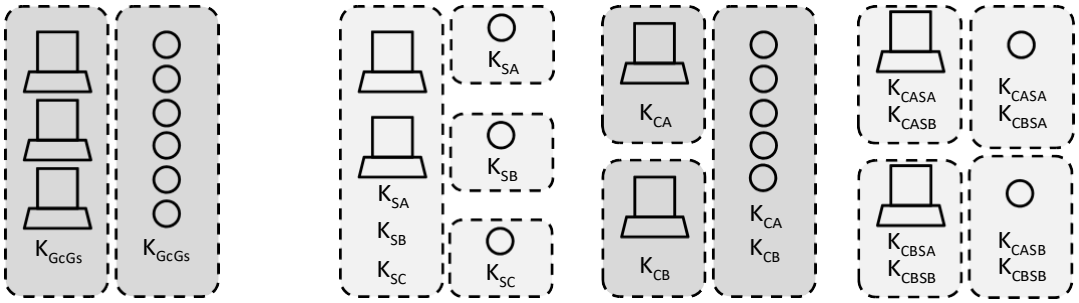

Figure 2: Approaches for sharing secret information

most key management systems are extensible (and in fact extensibility is an essential prop- erty for most applications), there are some mechanisms that do not provide extensibility. This is not advisable for IoT applications that have a growing number of clients and/or servers. As a result, this property must be considered in this analysis.

4. Analysis of Public Key Cryptography and Pre-shared Keys

In this section we will analyse the suitability of PKC and the pre-shared keys strategy to negotiate session keys between a sensor node and any external entities in an Internet context. Observe that it is possible to study different approaches for the pre-shared keys strategy. There is a very simple approach where all the members of the network share a common key (*1-1*). In addition, we can consider that all servers (*1-s*), all clients (*c-1*), or everyone (*c-s*) can have theirown unique keys. All these approaches (shown in Fig. 2) will be further explained and examinedin this section.

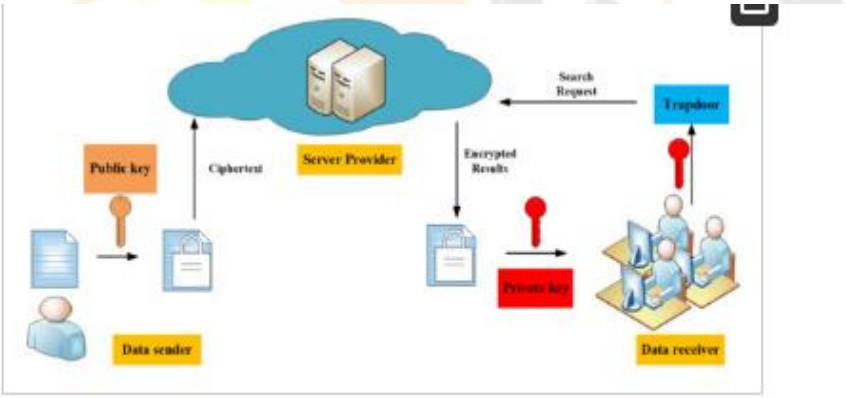

Figure 1. Model of public key encryption with keyword search (PEKS) system.

Public Key Cryptography

The development and implementation of public key primitives for sensor nodes by the aca- demic community, such as those based on Elliptic Curve Cryptography (ECC), has been a re- markable feat. The time to execute the main cryptographic operation of ECC, the scalar point multiplication, has been reduced from 34 seconds [31] in 2004 to less than 0.5 seconds [26] in 2009. With ECC, any node can make use of digital signature schemes (ECDSA), key exchange protocols (ECDH), and public key encryption schemes (ECIES) [32]. However, PKC is still too expensive to be used by sensor nodes implementing web servers, as the overhead of its software implementation (420 ms) is too high. Note that the use of other PKC primitives with extremely efficient encryption and verification operations such as Rabin [33] is discouraged. For example, in TLS server nodes need to decrypt a message encrypted by the client, but decryption in Rabinis basically as costly as a decryption operation in RSA.

In spite of this, the PKC strategy provides a very good set of properties. The distributionof the credentials can be done both offline and online, as the peer (public key) certificates can be preloaded in the devices or transmitted upon request. Also, both clients and servers canbe authenticated when using an adequate ciphersuite. The resilience is also high, because a subverted device usually stores only its public key and private key, thus an attacker can only impersonate that particular device. The scalability is not great but it is still good: while it is necessary to preload the certificate of a CA in order to verify all peer certificates signed by suchCA, those peer certificates can be exchanged during the negotiation process. Finally, as there is no limitations on the number of peer certificates that can be validated, the extensibility is excellent.

It seems that an Internet-connected sensor node that behaves as a server should not make useof PKC. However, the situation changes if

the sensor node behaves only as a client, accessing external web servers. Any external system cannot perform a DoS attack to the node by forcing it to calculate expensive PKC operations, as the node itself will make use of these operations only when it wants to open a secure connection with an external server. In terms of storage, a sensor node behaving as a client only needs the certificate of the CA that signed the certificate of the servers it accesses, and such certificate can even be retrieved online. Moreover, if the client node must authenticate himself, the only extra storage requirements are its public key and private key.

Analysis Results

Regarding the usage of PKC in server nodes, the performance of PKC in "Class II" nodes (4-8 Mhz, 4-16kB RAM, 48-256kB Flash) is too low to be used in a web server. However, there are some scenarios where the advantages of PKC, such as the ability to open a secure channel with a previously unknown entity, are essential for the successful development of real-world applications. For these particular cases, it is necessary to use more powerful nodes or hardware extensions (such as cryptographic chips) that can optimize the execution of PKC primitives.

As for client nodes, PKC is actually a viable option for applications without hard real-time requirements where its sensor nodes sporadically connect to external web servers, even for "Class II" nodes. Observe that there are other factors that must be taken into account when considering this solution, such as the energy consumption of the node, the code size of the PKC algorithms and their memory requirements, and the expected lifetime of the application.

Pre-shared keys

Table 1: Properties of PKC and PSK strategies

	PKC	Pre-shared key		
		1-1	*1-s*	*c-1*
Distribution	Online, Offline	Offline	Offline	Offline
Authentication	Client/Server	Group/Group	Group/Server	Client/Group
Overhead	ECC operations	Negligible	Negligible	Negligible
Resilience	High	Low/Low	Low/High	High/Low
Scalability	Good	High	High^{Server}	High^{Client}
Extensibility	Yes	Yes	Yes	Yes

In the pre-shared key strategy, the clients and the servers share some pre-established keying material that can be used to derive a shared key between peers. Actually, in a WSN scenario, there are multiple approaches to implement this particular strategy. For example, for external connections, a group of sensor nodes can have the same pre-shared secret (i.e. one key per WSN), or every sensor node can have its own pre-shared secret (i.e. n keys for a WSN of size n, one per sensor). The description of the different approaches is listed below, and Table 1 provides a summary of the properties of these approaches, alongside with the properties of the PKC approach.

1-1. In this approach, one group G_c of clients share the same pre-shared key ($k_{G_cG_s}$) with a group G_s of servers.

1-s. In this approach, every server s has its own pre-shared key (K_s). Consequently, clients must store one key per server they want to connect to.

c-1. In this approach, every client c has its own pre-shared key (K_c). As a result, the servers must know in advance which are the clients that will try to connect them, and store a key per client.

c-s. In this approach, every individual client c and server s share a common pre-shared key (K_{cs}). Note that this particular approach will not be analysed in the pre-shared key strategy, as it combines the disadvantages of both *1-s* and *c-1*.

One of the major drawbacks of these approaches is related to the distribution property. All secret information must be preloaded prior to any exchange of information, thus it is only possi-ble to securely connect clients and servers that already know of each other. Another drawback is the overall resilience of the approaches. As one element (clients in *1-s*, servers in *c-1*, everyone in *1-1*) will store all the pre-shared keys, such element is the weakest link of the security chain, and any adversary that gains control of that element will take control of the whole network. Be-sides, it is necessary to consider that the pre-shared key will be reused every time. Not only is advisable to use a mechanism that will derive a session key from the pre-shared key, but also such key should be updated over time.

Nevertheless, the benefits of using any of the pre-shared key approaches in a WSN context are still significant. The computational overhead is negligible, as the pre-shared key is already preloaded and there is no need to engage in costly negotiations. The scalability of all approaches is mostly high, although there are some cases (the *1-s* approach for client nodes and the *c-1* approach for server nodes) that are not advisable due to the limited memory available to the nodes. As for the authentication property, it is possible to provide either client authentication (*c-1*) or server authentication (*1-s*), although all approaches provide group authentication (e.g.if I have the pre-shared key then you know I can be trusted). Finally, the extensibility of all approaches is high, although there are some maintainability problems: in the *c-1* approach every server must be preloaded with the pre-shared key of any potential clients, and in the *1-s* approach whenever a server changes its key all its clients must also change it.

Analysis Results

For server nodes, the *1-s* approach is useful enough to think about deploying it in small real-world applications. Still, there are some factors that must be carefully considered, such as the maintenance of the pre-shared keys and the resilience of the clients. As for client nodes, the *1-s* approach is useful only if the number of external servers that must be contacted for a particular application is limited, due to memory constraints. Also, it should be mentioned that the *1-1* approach could be appropriate whenever it is important to assure the identity of a particular WSN, instead of the identity of specific server nodes. Finally, we give less importance to the *c-1* approach as it can be possible to implement a user authentication scheme in the server at a higher layer.

5. Analysis of WSN Key Management Systems

In recent years, the development of key management systems (KMS) for establishing link- layer keys between nodes in a WSN had been a very active research field [34]. It is possibleto classify these KMS protocols in four major frameworks: key pool framework, mathemat- ical framework, negotiation framework (which includes protocols that use pre-shared key ap- proaches), and public key framework. As we already discussed Public Key Cryptography and network-wide key protocols in sections 4.1 and 4.2, in this section we will focus on all the otherKMS frameworks. As our goal is to check whether some of the KMS protocols could be used tonegotiate session keys between remote entities in an Internet context, first we will describe these frameworks and we will review which KMS protocols could be suitable for fulfilling our goal. Afterwards, we will provide an analysis of the applicability of the selected KMS protocols.

Overview of KMS Frameworks

The key pool paradigm plays a pivotal role in the *key pool framework*, one of the most im- portant KMS frameworks ever proposed so far. The basic scheme behind this framework is quitesimple [35]. Firstly, the network designer creates a key pool, that is, a large set of precalculatedsecret keys. Secondly, before network deployment every node is assigned with a unique key chain, i.e., a small subset of the keys from the key pool. Thirdly, after the network deployment, the nodes exchange their identification numbers of the keys from their key chains, trying to finda common shared secret key. Finally, in case that two nodes do not share the same key, they tryto find a key path (i.e. a secure routing path) between them in order to negotiate a pairwise key. Note that the concept of a 'pool' can also be applied to other KMS of different frameworks.

The KMS protocols that make use of certain branches of mathematics (e.g. linear algebra, combinatorics, and algebraic geometry) to calculate link-layer keys belong to the *mathematicalframework*. Among the KMS protocols based on the linear algebra, the most important scheme is Blom scheme [36]. In this scheme, every node i can calculate the pairwise key it shares with another node j by solving $A(i)\ G(j)$, where G is a public Vandermonde matrix and A is calculated using a symmetric random secret matrix D. In the KMS protocols based on combinatorics, generalized quadrangle and symmetric design models [37] are the most widely used ones. Usinggeneralized quadrangles $GQ(s, t)$ or finite projective planes $FPP(q)$, a network designer can construct a key chain of size $s + 1$ or $q + 1$, respectively. Some of these key chains provide perfect connectivity, that is, every node share a key with any other node. Finally, for the KMS protocols developed based on algebraic geometry, the most well known scheme works based onthe bivariate polynomials [38]. In this scheme, by using a bivariate polynomial f, every node Ain a network is able to obtain a pairwise key with another node y by solving $f(A, y)$.

It is also possible to let the nodes negotiate their keys with their close neighbors right after the deployment of a WSN. All the protocols that generate their keys through mutual agreement be-long to the *negotiation framework*, and they usually work under the assumption that there is littleor no threat against the integrity of a WSN network in the very early stage of its lifetime [39]. Besides, all the KMS that consider the existence of a network-wide key [40] and those protocols that focus on organizing the network into dynamic or static clusters [41] also belong to this par-ticular framework. Notice that there are specific optimizations that can be done to the protocolsof this framework (such as code attestation and the Guy Fawkes protocol [42]) in order to assurethe authenticity of the peers in any stage of the network deployment.

Finally, every KMS protocol from the previously mentioned frameworks can be optimized inmultiple ways. For example, it is possible to use the knowledge of the final node locations in the

WSN field (i.e. deployment knowledge) to minimize the memory consumption and communica-tion overhead of the protocols. Other optimizations are focused on reducing the number and sizeof the messages that are used to discover/derive a common key. Also, there are some optimiza-tions that improve the underlying structure of a KMS in order to enhance certain properties suchas resilience and extensibility.

Suitability of KMS Protocols

Once we have presented the different KMS frameworks that can be used to establish a link-layer key between nodes, we need to select those KMS protocols that could be used to open a secure channel between remote entities. Such protocols must provide perfect key connectivity, guaranteeing the creation of a common key between any pair of nodes. In addition, they must not implement any optimizations that rely on the features of the deployment site or the wirelesschannel. Precisely, some of the KMS protocols that belong to the mathematical framework fulfillthese requirements. We will review such KMS protocols on section 5.3. On the other hand, mostof the KMS protocols are not suitable for the IoT context. We will justify this statement in the next paragraphs.

A major problem that affects the KMS protocols that belong to the key pool framework is the insufficient connectivity. Most key chains of the elements of the network are constructed byrandomly extracting a subset of keys from a key pool, where the size of the key chain is usuallymuch smaller than the size of the key pool. As a result, there is always a chance that two differentkey chains will not share a common key. This is not acceptable in an Internet scenario, wherea server node should receive any connections from the clients it knows, and viceversa. This weakness is shared by all the KMS protocols that make use of the key pool paradigm to improvetheir scalability while sacrificing their key connectivity. For example, some KMS protocols thatbelong to the mathematical framework use specially crafted key pools of random polynomials / matrices. This optimization improves their resilience, but sacrifices their key connectivity [38].

Other KMS protocols make use of the deployment knowledge to optimize the construction of their data structures. For example, many KMS protocols assume that the deployment region can be partitioned into small areas, known as cells. As an example of one of the optimizations, the key chains of nodes that belong to the same cell can be constructed in a deterministic way, rather than by selecting a random set from the key pool. This way, the connectivity between the members of the same cell (and even the members of other neighbouring cells in some cases)can be drastically improved [43]. However, this kind of deployment knowledge cannot be used in the IoT context. Client and server nodes are usually located in different physical locations, even in different countries. One possible approach to deal with this problem is to try to create a 'virtual' deployment knowledge, not based on the physical location of the devices but on their connections. However, in a client/server context, we would discover that all clients and serversthat are able to open

a secure channel with each other are in fact direct neighbours. In this 1-to-1situation, the optimizations of KMS protocols that make use of deployment knowledge cannot be applied.

As for those KMS protocols that belong to the negotiation framework, excluding those proto-
cols that make use of a network-wide key, they also can not be used in the IoT context. All such protocols make use of the wireless channel and its features to effectively negotiate a common key. Some protocols gradually increase the power of their transceivers to send information that will only be detected by its nearest neighbours. Other protocols make use of the limited range of the transceiver to establish a cluster and negotiate keys within that cluster, or use some mechanisms linked to the efficiency of the wireless channel (e.g. code attestation) in order to establish initial

trust. Such optimizations, that are based on the inherent nature of the wireless channel, cannot be used in the IoT context: client and server nodes usually belong to different networks, and they need to route the information through the Internet in order to be able to talk with each other.

Finally, combinatorics-based KMS protocols [37] are not considered in our analysis due to key connectivity and scalability / authentication issues. The KMS protocols based on general- ized quadrangles $GQ(s, t)$ do not provide perfect connectivity, thus they are not suitable for our purposes. As for the KMS protocols based on finite projective planes $FPP(q)$, the size of the per- fectly connected key chain increases with the size of the network: in networks of size $q^2 + q + 1$, where q is a prime number, the size of the key chain is $q + 1$. As a result, the communication overhead increases (two peers must exchange their key IDs in order to find a common key). Moreover, these protocols only provide group authentication, as it is not possible to distinguish a specific node from other nodes.

Analysis of Mathematical-based KMS Protocols

Table 2: Properties of the Blom scheme

	Blom			
	1-1	**1-s**	**c-1**	**c-s**
Distribution	Offline	Offline	Offline	Offline
Authentication	Group/Group	Group/Server^{OPT}	Client^{OPT} /Group	Client^{OPT} /Server^{OPT}
Overhead	VecMul(λ)	VecMul(λ)	VecMul(λ)	VecMul(λ)
Resilience	Low^{OPT} /Low^{OPT}	Low^{OPT} /High	High/Low^{OPT}	High/High
Scalability	High	High	High	High
Extensibility	Limited(n)	Limited(n)	Limited(n)	Limited(n)

Table 3: Properties of the Polynomial scheme

	Polynomial			
	1-1	**1-s**	**c-1**	**c-s**
Distribution	Offline	Offline	Offline	Offline
Authentication	Group/Group	Group/Server	Client/Group	Client/Server
Overhead	PolyEval(λ)	PolyEval(λ)	PolyEval(λ)	PolyEval(λ)
Resilience	Low/Low	Low/High	High/Low	High/High
Scalability	High	High	High	High
Extensibility	Yes	Yes	Yes	Yes

The KMS protocols that might be suitable for some IoT scenarios are the Blom scheme [36] and the Polynomial scheme [38]. A summary of the properties of these schemes, using the notation and the client/server combinations initially introduced in section 4.2, is shown in Table 2 and Table 3. First we will describe the foundations of these KMS protocols in detail, and later we will explain their main advantages and disadvantages.

The idea behind the Blom scheme is as follows: every node α (where α ID_{node_α}) receives a public vector g_α and a private vector a_α of size λ + 1. The public vector is a column from a(λ + 1) n Vandermonde matrix G, and the private vector is a row from a secret n (λ + 1) matrix A, where $A = (D \quad G)^T$ and D is a secret symmetric matrix of size (λ + 1) (λ + 1) over a finite field F_q. If two nodes i and j want to negotiate a common key, they simply need to exchange their public vectors g_i and g_j. After that, $key = g_i \quad a_j = g_j \quad a_i$. λ influences over the resilience of the protocol, as an adversary needs to capture λ + 1 nodes in order to derive all the secret keys. As for the other parameters, n defines the maximum number of nodes that can be used in the network, while q influences over the key size. Note that a public vector g_α can be generated from a single precalculated seed s^α. This optimization allows node authentication (if the nodes store s in advance), and reduces the space requirements and the communication

overhead. However, it increases the computational overhead (g_α must be generated from s^α) and decreases the resilience (any public vector g can be calculated from s^α and α) [36].

As for the Polynomial scheme (also known as Blundo scheme), each node α store a λ-degree polynomial share $f(\alpha, y) = f_\alpha^J(y)$, where $f(x, y)$ is a bivariate λ-degree polynomial $\sum_{j=0}^{\lambda} a_{ij} x^i y^j$ over a finite field F_q. If two nodes i and j want to negotiate a common key, they will exchange their unique IDs to obtain $key = f_i^J(j) = f_j^J(i)$. The resilience of this scheme is given by λ, as an adversary needs to retrieve λ + 1 polynomials in order to derive all the secret keys. Regarding q, not only it influences over the key size, but also limits the size of the node IDs: an ID must be
smaller than 2^q [38].

One of the major advantages of the Blom scheme in comparison with the pre-shared key strategy is the higher resilience. If the public vector g^α is not constructed with a seed s^α, any adversary that retrieves the public (g_α) and private (a_α) vectors of a certain α device of the network will not be able to impersonate other devices unless λ + 1 vectors are captured, even in the *1-1* approach.

Another benefit of both schemes is the higher scalability. In the Blom scheme a device only needs to store its public and private vectors g_α and a_α, while in the Polynomial scheme a device must store only a polynomial share $f_\alpha^J(y)$. The size of these vectors and functions is not given by the size of the network n, but by the size of the resilience factor λ. This factor can be tweaked in order to balance

the memory required by the schemes (and the size of the initialnegotiation message in case of the unoptimized Blom scheme) and the resilience of the network.However, there are some disadvantages that must be carefully considered. One problem is caused by the parameter q. In the Blom scheme, q influences on the key size, and in the Polyno-mial scheme q also influences over the size of the device IDs. Therefore, q should have a size ofat least 128 bits in order to support IPv6 addresses (as peer IDs) and key sizes of 128 bits. Thischoice affects the computational overhead of the schemes, because the keys need to be calculatedby multiplying two vectors (Blom scheme) or by evaluating a polynomial (Polynomial scheme)whose elements have 128 bits. And for 8-bit microcontrollers (with support for 16-bit operationssuch as the ATmega128L), these operations are expensive, requiring 64 word multiplicationsto multiply two 128 bits numbers using "schoolbook" multiplication. Note that the size of the resilience factor λ also influences over this computational overhead.

By using the information from the original papers and the datasheet from the ATmega128L microcontroller [44], we can estimate how long it takes for a sensor node to generate a shared key with another device (considering only addition and multiplication operations) with q = 128.One 128-bit multiplication operation takes 64 105 cycles, where 105 is the cost of multiplying two 16-bit numbers. On the other hand, one 128-bit addition operation takes 64 2 cycles, where2 is the cost of adding two 16-bit numbers.

Both the Blom scheme and the Polynomial scheme requires λ 128-bit multiplications and λ 128-bit additions, applying some optimizations like the Horner's rule for polynomial evalua- tion [45]. As a result, the time required to obtain a shared key with λ = 10 is approximately 67,360 cycles, or 8 ms in a 8 Mhz microcontroller. While the computational overhead of these schemes is quite low in comparison with the cost of a PKC operation, such overhead is not negligible and must be carefully considered in real-world implementations.

The Blom scheme has other problems, such as its extensibility. When constructing the G and A matrices, the application designer must calculate in advance the maximum number of peers n that will communicate with each other. As a result, the maximum number of clients andservers that will interact for a particular application is limited from the very beginning. Thisis not acceptable for large-scale applications. Another issue is related to the authentication of the devices. As the public vectors of the Blom scheme do not have any kind of certificate that

proves their authenticity, it is not possible to assure the identity of the devices, and any adversarycan launch a "man-in-the-middle" attack. Such attack can be avoided if the public vector $g_\alpha$is generated from the seed s^α, as every device can store s and use it to check the validity of a certain seed. Still, this check is very costly for sensor nodes. Using the binary exponentiation methods [46], a node needs to calculate no more than 2 128 128-bit multiplications, imposing an extra penalty of approximately 1,720,320 cycles (200 ms) using the values from the previousparagraph.

Analysis Results

One of the hurdles that might hinder the applicability of Mathematical KMS protocols in sensor servers is their computational overhead. However, if according to the requirements of theapplication a server node can afford such overhead, the polynomial scheme is a better choice than the pre-shared key strategy: the overall scalability is better, it is applicable for both client nodes and server nodes, and it is also possible to implement a *c-s* approach where both clients and servers can authenticate each other, although this *c-s* approach is not advisable if clients arenot trusted and can collaborate to obtain the original bivariate polynomial $f(x, y)$. Note that ifthe application requires more resilience and both extensibility and authentication are not crucial factors, it can be possible to use the Blom scheme. Finally, a drawback of both schemes that must be carefully considered is that two peers always generate the same session key. Therefore,the use of random values during the key generation process should be considered.

6. Conclusions and Future Work

In this paper we have studied possible solutions for the problem of establishing a session key between a client and a server in the context of the Internet of Things, where one or more peers are sensor nodes from a Wireless Sensor Network. This problem is not trivial, as the inher- ent limitations of many sensor nodes might hinder the application of existing key management mechanisms. In our analysis, we also have tried to apply existing Key Management Systems specialized in establishing link-layer keys between neighbouring nodes, as their underlying goalis similar to those mechanisms that try to establish keys between remote entities. Our results canbe summarized as follows:

In server nodes, Public Key Cryptography requires of powerful nodes and/or cryptographic chips. In client nodes whose applications only need to connect external servers from timeto time, PKC could be a viable solution.

Pre-shared key approaches can be useful for server nodes in small real-world applications.However, existing mathematical-based KMS, like the Polynomial scheme, provide betterproperties if the application can afford the extra overhead. Observe that such KMS can also be a viable solution for client nodes.

The analysis presented in this paper is by no means a complete study on all the possible sensor network scenarios that we can find in the Internet of Things, as we have only considered the existence of client/server connections with no trusted third parties. If we take into account otherscenarios such as P2P scenarios, then it might be possible to make use of other key managementmechanisms, such as those systems based on the key pool paradigm. Moreover, the existence of a trusted third party might enable the online distribution of specific credentials (e.g. shared keys, Blom key pairs, polynomial shares) to trusted clients. Such scenarios will be considered infuture works on this area.

References

[1] INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems, in co-operation with the Workinggroup RFID of the ETP EPOSS, Internet of things in 2020: Roadmap for the future, 27 May 2008.

[2] V. Ovidiu, M. Harrison, H. Vogt, K. Kalaboukas, M. Tomasella, K. Wouters, S. Gusmeroli, S. Haller, Internet of Things Strategic Research Roadmap, European Commission - Information Society and Media DG, 2009.

[3] National Intelligence Council, Disruptive civil technologies - six technologies with potential impacts on us interestsout to 2025, Conference Report CR 2008-07, april 2008.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: A survey, Computer Networks38 (4) (2002) 393–422.
[5] Wireless Systems for Automation, http://www.isa.org/, Accessed on October, 2010.
[6] IBM: A Smarter Planet, http://www.ibm.com/smarterplanet/, Accessed on October 2010.
[7] HP: A Central Nervous System for the Earth, http://www.hpl.hp.com/news/2009/oct-dec/cense.html,Accessed on October 2010.
[8] G. Montenegro, N. Kushalnagar, J. Hui, D. Culler, RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4Networks (2007).
[9] Routing Over Low power and Lossy networks (roll) Working Group. Internet Engineering Task Force (IETF), https://datatracker.ietf.org/wg/roll/charter/, Accessed on October 2010.
[10]Constrained RESTful Environments (core) Working Group. Internet Engineering Task Force (IETF), https:// datatracker.ietf.org/wg/core/charter/, Accessed on October 2010.
[11]D. Guinard, M. Fischer, V. Trifa, Sharing Using Social Networks in a composable Web of Things, in: 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM 2010), 2010, pp. 702–707. doi:10.1109/PERCOMW.2010.5470524.
[12]Libelium: Interfacing the Sensor Networks with the Web 2.0, http://www.libelium.com/, Accessed on Octo-ber 2010.
[13]R. Roman, J. Lopez, Integrating wireless sensor networks and the internet: A security analysis, Internet Research 19 (2) (2009) 246–259. doi:10.1108/10662240910952373.
[14]D. Christin, A. Reinhardt, P. Mogre, R. Steinmetz, Wireless Sensor Networks and the Internet of Things: SelectedChallenges, Proceedings of the 8th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze", 2009.
[15]C. Alcaraz, J. Lopez, A Security Analysis for Wireless Sensor Mesh Networks in Highly Critical Systems, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40 (4) (2010) 419–428. doi:10.1109/TSMCC.2010.2045373.
[16]M. Healy, T. Newe, E. Lewis, Analysis of Hardware Encryption Versus Software Encryption on Wireless Sensor Network Motes, in: S. C. Mukhopadhyay, G. S. Gupta (Eds.), Smart Sensors and Sensing Technology, Vol. 20of Lecture Notes in Electrical Engineering, Springer Berlin Heidelberg, 2008, pp. 3–14. doi:10.1007/978-3-540- 79590-2 1.
[17]A. Francillon, C. Castelluccia, TinyRNG: A Cryptographic Random Number Generator for Wireless Sensors Net-work Nodes, in: 5th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt 2007), 2007, pp. 1–7. doi:10.1109/WIOPT.2007.4480051.
[18]N. Kushalnagar, G. Montenegro, C. Schumacher, RFC 4919: IPv6 over Low-Power Wireless Personal Area Net- works (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals (2007).
[19]T. Dierks, E. Rescorla, RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2 (2008).
[20]OASIS Web Services Security (WSS) TC, http://www.oasis-open.org/committees/wss/, Accessed onOctober 2010.
[21]S. Amin, Y. J. Young, M. Siddiqui, C. S. Hong, A novel Intrusion Detection Framework for IP-based sensor networks, in: International Conference on Information Networking (ICOIN 2009), 2009, pp. 1–3.
[22]R. H. Weber, Internet of Things - New security and privacy challenges, Computer Law & Security Review 26 (1)(2010) 23–30. doi:10.1016/j.clsr.2009.11.008.
[23]P. Eronen, H. Tschofenig, RFC 4279: Pre-Shared Key Ciphersuites for Transport Layer Security (TLS) (2005).
[24]E. Rescorla, N. Modadugu, RFC 4347: Datagram Transport Layer Security (2006).
[25]OASIS Web Services Secure Exchange (WS-SX) TC, http://www.oasis-open.org/committees/ws-sx/, Accessed on October 2010.
[26]L. B. Oliveira, D. F. Aranha, C. P. Gouvea, M. Scott, D. F. Camara, J. Lopez, R. Dahab, TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks, Computer Communications In Press, Corrected Proof (2010) –. doi:10.1016/j.comcom.2010.05.013.
[27]G. Bianchi, A. T. Capossele, A. Mei, C. Petrioli, Flexible key exchange negotiation for wireless sen- sor networks, in: Proceedings of the fifth ACM international workshop on Wireless network testbeds, ex- perimental evaluation and characterization, WiNTECH '10, ACM, New York, NY, USA, 2010, pp. 55–62. doi:http://doi.acm.org/10.1145/1860079.1860090.
[28]I. F. Akyildiz, D. Pompili, T. Melodia, Underwater Acoustic Sensor Networks: Research Challenges, Ad Hoc Networks 3 (3) (2005) 257–279. doi:10.1016/j.adhoc.2005.01.004.
[29]R. D. Pietro, L. Mancini, C. Soriente, A. Spognardi, G. Tsudik, Catch Me (If You Can): Data Survival in Unattended Sensor Networks, in: 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008), 2008, pp. 185–194. doi:10.1109/PERCOM.2008.31.
[30]M. Johnson, M. Healy, P. van de Ven, M. Hayes, J. Nelson, T. Newe, E. Lewis, A Comparative Re- view of Wireless Sensor Network Mote Technologies, in: IEEE Sensors 2009, 2009, pp. 1439–1442. doi:10.1109/ICSENS.2009.5398442.
[31]D. Malan, M. Welsh, M. Smith, A Public-key Infrastructure for Key Distribution in TinyOS based on Elliptic Curve Cryptography, in: 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communicationsand Networks (IEEE SECON 2004), 2004, pp. 71–80. doi:10.1109/SAHCN.2004.1381904.
[32]A. Liu, P. Ning, TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks, in: International Conference on Information Processing in Sensor Networks (IPSN 2008), 2008, pp. 245–256. doi:10.1109/IPSN.2008.47.
[33]M. Rabin, Digitalized signatures and public-key functions as intractable as factorization, Tech. rep., Cambridge, MA, USA (1979).
[34]M. A. S. Jr., P. S. Barreto, C. B. Margi, T. C. Carvalho, A Survey on Key Management Mechanisms for Distributed Wireless Sensor Networks, Computer Networks 54 (15) (2010) 2591–2612. doi:10.1016/j.comnet.2010.04.010.
[35]L. Eschenauer, V. D. Gligor, A Key-Management Scheme for Distributed Sensor Networks, in: 9th ACM confer-ence on Computer and Communications Security (CCS 2002), 2002, pp. 41–47. doi:10.1145/586110.586117.
[36]W. Du, J. Deng, Y. S. Han, P. Varshney, J. Katz, A. Khalili, A Pairwise Key Predistribution Scheme for Wire- less Sensor Networks, ACM Transactions on Information and System Security (TISSEC) 8 (2) (2005) 228–258. doi:10.1145/1065545.1065548.
[37]S. A. Camtepe, B. Yener, Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks, IEEE/ACM Transactions on Networking 15 (2) (2007) 346–358. doi:10.1109/TNET.2007.892879.
[38]D. Liu, P. Ning, R. Li, Establishing Pairwise Keys in Distributed Sensor Networks, ACM Transactions on Informa-tion and System Security 8 (1) (2005) 41–77. doi:10.1145/1053283.1053287.
[39]R. Anderson, H. Chan, A. Perrig, Key infection: Smart Trust for Smart Dust, in: Proceedings of the 12th IEEE In-ternational Conference on Network Protocols (ICNP 2004), 2004, pp. 206–215. doi:10.1109/ICNP.2004.1348111.
[40]B. Lai, S. Kim, I. Verbauwhede, Scalable Session Key Construction Protocol for Wireless Sensor Networks, in: IEEE Workshop on Large Scale Real-Time and Embedded Systems (LARTES 2002), 2002.
[41]B. Panja, S. Madria, B. Bhargava, Energy and Communication Efficient Group Key Management Protocol for Hier-archical Sensor Networks, in: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous,and Trustworthy Computing (SUTC'06), Vol. 1, 2006, p. 8 pp. doi:10.1109/SUTC.2006.1636204.
[42]A. Seshadri, M. Luk, A. Perrig, SAKE: Software Attestation for Key Establishment in Sensor Networks, Ad Hoc Networks In Press, Corrected Proof. doi:10.1016/j.adhoc.2010.08.011.
[43]J. Jaworski, M. Ren, K. Rybarczyk, Random Key Predistribution for Wireless Sensor Networks Using DeploymentKnowledge, Computing 85 (1–2) (2009) 57–76.
[44]ATmega128, AVR Solutions, http://www.atmel.com/dyn/products/product_card.asp?part_id=2018, Accessed on October 2010.
[45]W. Dorn, Generalizations of Horner's Rule for Polynomial Evaluation, IBM Journal of Research and Development6 (2) (1962) 239–245. doi:10.1147/rd.62.0239.
[46]A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2001.URL http://www.cacr.math.uwaterloo.ca/hac/