Key Management
for
Wireless Ad hoc Networks

Von der Fakultät für Mathematik, Naturwissenschaften und Informatik
der Brandenburgischen Technischen Universität Cottbus

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

vorgelegt von

Telekommunication Enginieur

David Sánchez Sánchez

geboren am 26–02–1978 in Terrassa (Spanien)

Gutachter: Prof. Dr-Ing. Rolf Kraemer

Gutachter: Prof. Dr-Ing. Jörg Nolte

Gutachter: Dr.-Ing. Jordi Forné Muñoz

Tag der mündliche Prüfung: 29-06-2006

to my love Sole

Zusammenfassung

Schlüsselmanagement ist ein grundlegender Sicherheitsdienst, um sichere drahtlose Ad Hoc Netze (WAHN) zu ermöglichen. Bestehende Schlüsselmanagementlösungen, basierend entweder auf Public Key Infrastructure (PKI) oder auf Key Pre-distribution Scheme (KPS), haben Beschränkungen für WAHNs.

Wir entwickeln erstens die *Hybrid Key Management Infrastructure* (HKMI) für WAHNs, welche die PKI um Trust- und Kooperationsprotokolle ergänzt, um eine leistungsfähige Sicherheitslösung zu konstruieren.

Wir entwickeln zweitens das *Deterministic Pairwise Key Pre-Distribution Scheme* (DPKPS) für großangelegte dynamische WAHNs, die aus Geräten mit beschränkten Ressourcen bestehen. Das DPKPS verwendet ein kombinatorisches Design für die Vorverteilung mehrerer zweidimensionaler polynomieller Anteile an die WAHN-Knoten.

Zukünftige Arbeit umfasst die weitere Verbesserung der Widerstandsfähigkeit des DPKPS, die Entwicklung einer Schlüsselmanagementinfrastruktur auf der Grundlage von DPKPS, den Entwurf DPKPS-basierter Zugriffskontrollsysteme, und die Integration der HKMI mit DPKPS in eine einheitliche Schlüsselmanagementarchitektur.

Abstract

Key management is a fundamental security service to enable secure wireless ad hoc networks (WAHN). To date existing key management solutions based on either public key infrastructures (PKI) or key pre-distribution scheme (KPS) exhibit limitations for WAHNs.

We firstly develop the *Hybrid Key Management Infrastructure* (HKMI) for WAHNs composed of moderate-resource devices. The HKMI complements PKI with trust and cooperation protocols to construct an performance efficient security solution.

We secondly develop the *Deterministic Pairwise Key Pre-Distribution Scheme* (DPKPS) for large-scale dynamic WAHNs composed of low-resource devices. The DPKPS applies a combinatorial design for the pre-distribution of multiple bivariate polynomial shares to WAHN nodes.

Future work comprises further improving the resiliency of the DPKPS, completing a key management infrastructure on the basis of the DPKPS, the design of DPKPS-based access control mechanisms, and the integration of the HKMI with the DPKPS in a unified key management architecture.

Summary

Wireless ad hoc networking is the enabling technology for paramount civilian and military applications requiring easy and quick (and often unmanned and inexpensive) network deployments. Typical (or foreseen) wireless ad hoc network (WAHN) applications include disaster recovery, military operation, environment monitoring, patient care and patient vital sign monitoring and others.

A WAHN is a collection of autonomous nodes that communicate with each other by forming a single-hop or, often, multi-hop wireless network and by maintaining connectivity in a decentralized manner. The network topology is in general dynamic, because the connectivity among the nodes varies with itinerant, quitting and joining nodes.

Security is essential for WAHNs. Security services protect the confidentiality, the integrity and the authenticity of communications from unauthorized parties attempting on legitimate WAHN communications.

Key management is a fundamental security service, which, by providing and managing the basic cryptographic keying material, fundaments security services preserving confidentiality, integrity and authenticity.

The design of key management mechanisms for WAHNs is a particularly complex issue. Firstly, because of the lack of an infrastructure (e.g. dedicated servers), WAHNs require self-organized key management protocols. Secondly, in order to maximize WAHN longevity, because nodes typically run on batteries, energy efficiency is a strict requirement in the design of key management mechanisms and protocols. Thirdly, WAHN scalability, membership dynamics and sudden changes on network topology must be also contemplated in the design of a performance-aware key management service. Finally, the operational requirements and the use model of WAHN applications need to be considered to design consistent key management systems.

To date there exist two relevant key management concepts for WAHNs: public key infrastructure (PKI) and key pre-distribution scheme (KPS). PKIs can be applied to dynamic WAHNs of powerful computing nodes (such as Laptops). However, the expensive (both computationally and timely) modular exponentiations required by public key operations and the excessive length of messages exchanged by public key-based protocols hinders successful deployment of PKI-only based security for WAHNs of moderate-resource devices (such as mobile phones, PDAs and other embedded computing systems).

In low-bandwidth large-scale WAHNs of low-cost low-resource nodes, public key cryptography is strictly prohibited and cooperative security is neither robust nor scalable. In this context, to date key management systems base on KPSs because key pre-distribution enables nodes to establish lightweight[1] symmetric keys without online trusted server support.

---

[1] With associated energy and time efficiency properties.

The majority of the existing KPSs for large-scale WAHNs relax KPS connectivity for resiliency and network size scalability and, thus, cannot generally be applied to dynamic WAHNs. Additionally, these schemes make use of third-party collaborative protocols, which imply a significant reduction of energy efficiency and security robustness. On the other hand, existing KPSs enabling direct key establishment solve the third-party involved problems but fail to provide authentication in an performance-aware manner.

In the first part of this thesis we solve the limitations of PKI in the context of WAHNs composed of moderate-resource devices. Our solution, the *Hybrid Key Management Infrastructure* (HKMI), complements PKI with trust and cooperation protocols to construct a much more performance efficient (in terms of time and energy efficiency) security solution.

Authenticated key establishment is a key process for securing the channel between two communicating parties. PKI-based key establishment involves exchanging bulky[2] public key certificates (typically exceeding 2 Kbits) and expensive public key operations (a single RSA digital signature takes around 83[3] ms. in a moderate 206 MHz processor). The results in this thesis show that HKMI can improve the energy consumption and the time delay incurred by PKI-based key establishment in up to 3 orders of magnitude. This result is of particular relevance for (low bandwidth) WAHNs of moderate-resource devices, particularly in applications with strict maximum energy consumption and delay restrictions.

The trade off between security and performance in HKMI is positive, particularly in applications with low risk of misbehaving users. However, for WAHN applications with a potential risk for node compromise or misbehavior, our solution inherits the well-known security vulnerabilities of trust and cooperation security protocols. Nonetheless, a robust security architecture for such applications requires a node reputation mechanism, which can also be employed to enhance the security strength of HKMI.

In the second part of this thesis we solve the limitations of KPSs for large-scale dynamic WAHNs composed of low-resource devices. We present the *Deterministic Pairwise Key Pre-Distribution Scheme* (DPKPS), which originally applies a combinatorial design for the pre-distribution of multiple bivariate polynomial shares to WAHN nodes. The DPKPS is, to date, to the best of our knowledge the only performance-aware KPS that enables direct[4] authenticated key establishment.

The parameters of the DPKPS can be tuned to best-fit the security and performance (in terms of energy and time efficiency as well as of number of accommodated nodes)

---

[2] Wireless is a shared a media. Consequently, effective user throughput is mainly restricted by the number of active WAHN members (may involve from tens to thousands of nodes) and the capacity of the WAHN enabling wireless technology (A maximum of 250 Kbps for ZigBee™/082.15.4 or of 1Mbps for Bluetooth® v1.0).

[3] Some applications, such as medical, have strict timing requirements on connectivity establishment. A representative target value is in the order of a few milliseconds.

[4] Just involving the two communicating parties.

requirements of a particular WAHN application. The DPKPS is equally applicable to small, medium or large size WAHNs of resource-constrained fixed or mobile nodes. For instance, the DPKPS can be used to accommodate up to 3,997,696 nodes. In such a case, each node needs to store just 1 Kbytes of keying material. Two nodes can establish a key by merely exchanging 38 bits of data and investing a negligible quantity of their battery resource. The DPKPS resists exposure of the keying material when nodes are captured. In our example, the attacker needs to *selectively* attack up to 125 nodes to break the security of the distributed keying material. For an attacker who captures nodes randomly, breaking the DPKPS results even harder.

We successfully proved the performance and the practical feasibility of the DPKPS in a demonstrator for body sensor networking. Sensor nodes authenticate and establish secure communications transparently and unobtrusively to the human user and without apparent delays.

We propose four lines to continue this work. Firstly, future work may concentrate on further improving the resiliency of the DPKPS. A second line may contribute on completing a key management infrastructure on the basis of the DPKPS. Thirdly, the design of DPKPS-based access control mechanisms for mobile sensor networking, including role, group and user access control. Finally, a final task may explore the integration of the HKMI with the DPKPS in a unified key management architecture for WAHNs.

Acknowledgements

Developing this PhD has been one of the most challenging and fascinating experiences and achievements I have ever lived. The success of my work has been possible thanks too a big number of great people and institutions.

Thanks to my love Sole for encouraging and supporting me throughout the whole development of this thesis. Equally, I feel strong gratitude to my relatives and friends in Spain who where always with me regardless the long distance.

Thanks to my friends who enjoyed with me plenty of funny experiences and gave me warming support to develop my PhD in Aachen. I particularly want to mention Javier Espina, Francesco Gallo, Daniel Totev, Barbara Leung, Toni Lopez and Yoli Martori, Raul Duran, Mark Carpaij, Marco Ruffini, Caroline Souvestre, Zhenia Trusova, Alice, Joan, Alvaro Chena, Colm, Emilio Farras, Eva Lasarcyk, Fabiana, Giorgio, Helko Lehmann, Jordanne Besnard, Oscar García, Salvador Boleko, Marta Millán, Nazari Meca, Muriell Méen, Natalia Rojo, Patricia Folgueras, Peter Zink, Samuel Osorio, Silvia Prieto and Steven Wong.

Thanks to Philips Forschungslabor in Aachen and the scientists of the Connectivity Systems group. Particular gratitude to Dr. Heribert Baldus and Dr. Martin Elixmann for continuously following and supporting my work. Thanks also to Francesc Dalmases, Dr. Klaus Weidenhaupt, Thomas Falck, Dr. Henning Maass, Mr. H. Gapisch and. Karin Klabunde as inestimable references in several aspects of my thesis.

Thanks to the Marie Curie Fellowship commission for financially supporting this thesis.

Thanks to Prof. Dr. Rolf Kraemer for academically supporting this thesis. Thanks also to the Cottbus Universität and Lehrstuhl für Systeme.

Table of Contents[5]

[5] You can see a more detailed Table of Contents at the end of this Thesis.

1 Introduction and Motivation

Security is essential for wireless ad hoc networks (WAHN). Security services protect the confidentiality, the integrity and the authenticity of communications from unauthorized parties attempting on legitimate WAHN communications. *Key management* is an indispensable security service, that, by providing and managing the basic cryptographic keying material, fundaments further security services preserving confidentiality, integrity and authenticity [31].

Wireless ad hoc networking is the enabling technology for paramount civilian and military applications requiring easy and quick (and often unmanned and cheap) network deployments. Typical (or foreseen) WAHN applications include disaster recovery, military operation, environment monitoring, patient care and patient vital sign monitoring and others. For instance, a WAHN of medical wireless portable actuators (including respirators, infusion pumps, etc.) can be employed to care a patient in an intensive care unit. Similarly, a WAHN of wearable wireless vital sign sensors (including electrocardiogram, blood pressure, etc.) and mobile wireless display devices (such as portable monitors and PDAs) can be employed to monitor patient health in the hospital.

Formally, a wireless ad hoc network (WAHN) is a collection of autonomous nodes that communicate with each other by forming a single-hop or, often, multi-hop wireless network and by maintaining connectivity in a decentralized manner. Typically, WAHN links have less bandwidth than links of a wired network. Each node in a WAHN may work both as a host and a router, and the control of the network is distributed among the nodes. The network topology is in general dynamic, because the connectivity among the nodes may vary with itinerant, quitting and joining nodes [42].

Wireless technologies such as IEEE 802.11 and Bluetooth® are increasingly being used for security and privacy-sensitive individual, commercial and industrial WAHN applications. Early developments on security for wireless technologies were completely flawed with security vulnerabilities [43][44] and simple and successful attacks[6] have been reported (potentially) compromising the security of corporate networks and the privacy of individuals. Poor wireless security designs may not only degrade user trust on wireless or generate tremendous vendor losses but, since wireless starts to be employed in safety-sensitive applications such as medical, they may also crucially affect human lives. For instance, consider the effect of an intruder wirelessly switching your respirator off, or the effect of somebody wirelessly modifying your electrocardiogram from a normal to a flat signal during an operation.

IEEE reacted in June 2004 by approving the security standard IEEE 802.11i, which aims at mending previous flaws of 802.11 particularly for wireless access to infrastructure networks. ZigBee™, a new emerging wireless technology for low-cost low-power nodes, is specifying lightweight security for simple WAHN applications [46]. However, WAHN applications requiring performance-aware security solutions remain yet uncovered by 802.11i, Bluetooth and ZigBee.

[6] Just try searching for “WLAN hacking” or “WLAN security” in your favorite Internet search engine.

Because of its central role in security, research on key management for WAHNs, particularly on key initialization and establishment, is an active research area. For instance, the pioneering work of Stajano and Anderson [46] analyzed and elegantly solved the problem of bootstrapping security between devices of small-scale WAHNs. Subsequently, there has also been a generous bunch of excellent proposals to adapt public key infrastructures (PKI) to WAHNs [6]-[15]. In the recent years, because of the boom of sensor networks, key pre-distribution schemes (KPS) aim at solving the problem of bootstrapping security in large-scale WAHNs of static resource-constrained nodes [20]-[30]. Nowadays, researchers are solving the same problem in large-scale WAHNs with tight operational requirements and node resource constraints.

We originally included the problems of mobility, network dynamics and partitions to the design of a consistent KPS [2][3]. This fills the gaps left by previous KPS proposals, which broadly considered large-scale low-resource WAHNs to be static and wide-connected.

The design of key management mechanisms for WAHNs is a particularly complex issue. Because of the lack of an infrastructure (e.g. dedicated servers) supporting such an essential security service, WAHNs require self-organized and, often, cooperative key management protocols. A cooperative design, in turn, demands to thoroughly update the models of trust and threat previously considered for infrastructure-based networks.

In order to maximize WAHN longevity (e.g. when nodes are deployed in remote areas), because nodes typically run on batteries, energy efficiency is a strict requirement in the design of any WAHN system and protocol, particularly, of key management mechanisms and protocols. In some cases a cooperative key management service can help reducing energy cost of establishing keys (see Chapter 3). In other cases, direct key establishment results in a much more energy preserving solution (see Chapter 4).

A WAHN may be composed of tens, hundreds or thousands of nodes. In typical WAHN applications, nodes move autonomously and may, therefore, continually join or quit the WAHN. Additionally, the WAHN may divide in two or more subsets of nodes leading to, consequently, isolated nodes and/or subsequently unavailable network services. Thus, WAHN scalability, membership dynamics and sudden changes on network topology must also be considered to design a consistent performance-aware key management service.

Finally, the operational requirements and the use model of WAHN applications need to be considered to design corresponding and satisfying security. For instance, in applications where the devices belong to the same administration and are carried or worn by human users, cooperative security may be used to design performance-aware security protocols (see Chapter 3). Conversely, in applications where nodes are left unattended in public places, an uncooperative security design minimizes the vulnerabilities arising from exposed nodes (see Chapter 4).

Our thesis is that in WAHNs the most adequate key management service enables any pair of nodes to establish a key independently of any dedicated security server and in a performance-aware fashion, i.e. it is flexible to WAHN dynamics and fulfills WAHN energy efficiency requirements. In this thesis, we mainly focus on key initialization and establishment in WAHNs.

In our research for key management mechanisms for WAHNs we find PKI and KPS the two main concepts in line with our thesis, which we extensively revise and evaluate in Chapter 2.

PKIs can be applied to dynamic WAHNs of moderate (or powerful) computing nodes. However, because public key operations demand computationally expensive modular exponentiation and because messages exchanged by public key-based protocols are excessively large, PKIs do not result in an optimized energy efficient solution in WAHNs (see Chapters 2 and 3).

In Chapter 3, our *Hybrid Key Management Infrastructure* (HKMI) exploits cooperative trust to solve the energy limitations of PKIs in WAHNs of moderate computing nodes.

In low-bandwidth WAHNs of inexpensive (typically no tamper-resistant) and resource constrained (little memory space, low CPU power and short battery life) nodes, public key cryptography is prohibited and cooperative security is neither robust nor scalable (see Chapters 2 and 4). In this context, because KPSs enable nodes to establish lightweight symmetric keys without online trusted server support, to date KPS-based security emerges as the best alternative. However, because existing KPSs for large-scale WAHNs typically relax KPS connectivity for resiliency and network size scalability, they cannot generally be applied to dynamic WAHNs (see Chapter 2).

In Chapter 4, our *Deterministic Pairwise Key Pre-Distribution Scheme* (DPKPS), especially conceived for large-scale WAHNs of inexpensive and resource constrained mobile nodes, is, to the best of our knowledge, the first KPS proposal for WAHNs designed with independence of network connectivity and node density assumptions. It is thus equally applicable to dynamic WAHNs of small, medium or large size.

The rest of the thesis is organized as follows. Chapter 2 thoroughly reviews and evaluates PKIs and KPSs in the WAHN domain. We present and evaluate our solution to improve WAHN PKIs in Chapter 3. Our novel DPKPS is presented, evaluated and implemented in Chapter 4. Chapter 5 shows a DPKPS demonstrator for medical body sensor networks. Finally, we conclude this thesis and draw future lines of work in Chapter 6.

2 Key Management Approaches for WAHNs

In this section, we review and evaluate the literature on key management for WAHNs. Further assessments on key management approaches for WAHNs can also be found in [1][2][3]. Other surveys on a broad range of security aspects for WAHNs can be found in [30][38][39][40][41].

To fully appreciate the contents of this chapter we strongly recommend the reader to get familiar with the basics of cryptography and network security. For a simple yet excellent introduction to cryptography we refer the reader to [35]. For a broader coverage yet structured perspective on security we refer the reader to [16]. For a deep and scientific view on cryptography and security refer to the ever masterpiece, the Handbook of Applied Cryptography [31]. We also suggest a primer on the idiosyncrasies of ad hoc and sensor networking [36][37].

In section 2.1 we briefly introduce the concept and objectives of key management. Section 2.2 introduces the requirements imposed by WAHNs on key management. In section 2.3, we review server based key management concepts. In Section 2.4, we evaluate public key infrastructures for WAHNs. Section 2.5 reviews and evaluates key pre-distribution schemes. The summary and conclusions of this chapter are presented in Section 2.6.

2.1 Key Management

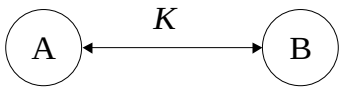
Key management is defined as "the set of techniques and procedures supporting key establishment and the maintenance of keying relationships between authorized parties". Key management sets the fundament for securing cryptographic techniques providing data confidentiality and integrity, entity and data authentication, and digital signatures [31].

In a generic communication system where two or more parties may communicate, key management implements (a set of) the following processes and mechanisms:

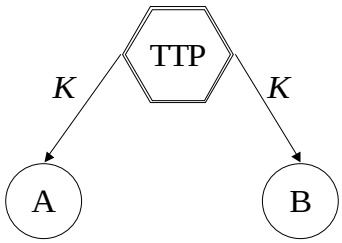
(a) Initialization of system users within a security domain (e.g. by registering a user identity in a security server),
(b) Generation, distribution and installation of keying material (e.g. loading of the symmetric key or the public/private key pair and corresponding public key certificate of a user into the user's computer),
(c) Controlling the use of keying material (e.g. intended cryptographic use of a key),
(d) Update, revocation and destruction of keying material,
(e) Storage, backup/recovery and archival of keying material.

Key establishment is "any process whereby a shared secret key becomes available to two or more parties" [31]. Assuming two parties *A* and *B* initialized in a security domain and provided with initial keying material, there are two models of key establishment:

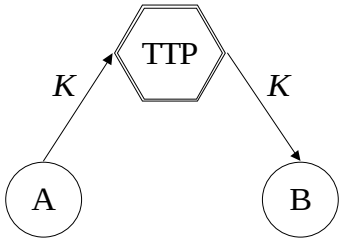
1. *Direct*. Both parties establish a key communicating directly.



2. *Centralized*. A trusted third party (TTP) generates and distributes a key to both parties.



3. *Translated*. Firstly, one of the parties, *A* or *B*, generates and distributes a key to the TTP. Then, the TTP forwards the key to the other party, *B* or *A*.



Direct key establishment is enabled by KPSs and by PKIs. In KPSs, users are *initially* provided with one or a set of shared *symmetric* keys. One (or a subset) of keys can be employed to establish a key in a process conforming to model (1). In PKIs, typically users are initially provided with a *public*/*private* key pair and a *public key certificate*. The public/private key pair and the certificate can be employed to establish a key in a process conforming to model (1).

In centralized key management each user is *initially* provided with an exclusive *long-term symmetric* key shared with the TTP. The long-term symmetric key is used to establish a key in a process conforming to model (2).

Translated key management is an extension of model (1). In this model, *A* and *B* may not directly share a common key. Therefore, they need to use TTP as an intermediary to establish a key. Alternatively, *A* and *B* may share keying material enabling direct key establishment. However, they rather use the TTP as an intermediary.

The advantages of centralized key management are threefold. Firstly, each user only needs to store a long-term symmetric key. This property is of particular relevance in applications with devices with scarce memory resources. Secondly, it enables simple key control and revocation. Thirdly, keying material of a user *A* is totally uncorrelated with keying material of any other system user. This property is of particular relevance in applications with no tamper-resistant devices.

Nonetheless, the following disadvantages of centralized key management hinder their successful deployment for WAHNs:

- The TTP needs to be online during the key establishment process,
- potential loss of communication system security, if TTP is compromised,
- performance bottleneck, if TTP is overloaded with requests,
- loss of service availability, if TTP fails.

Decentralized key management, enabled by PKIs or KPSs, solves the limitations of the centralized approach. The key establishment processes can be executed directly, i.e. independently from TTPs.

However, traditional public key cryptography demands exponential modular operations with large numbers. These operations are unfeasible in strict resource constrained devices with extremely low-power CPU and little RAM space. Additionally, because the length of public keys and certificates is over the 1 Kbytes, public key-based protocol messages get excessively large for low-cost low-bandwidth WAHNs.

KPSs appear to date as an interesting alternative to PKI for low-cost low-bandwidth WAHNs. However, existing KPSs do not scale to a large number of users unless security trade-offs are accepted in the target application.

A last issue of decentralized key management, where no TTP holds the responsibility for system security, is key control and revocation.

In the recent years there have been a great number of research proposals adapting PKIs and KPSs to WAHNs. Thus, the state-of-the-art of key management has evolved considerably. We extensively review and evaluate the related literature in the rest of the chapter. Later in chapters 3 and 4 we will describe our own proposals to address and solve some of the performance limitations of PKIs and KPSs in WAHNs, respectively.

2.2 Requirements of Key Management for WAHNs and Evaluation Criteria

The ad hoc nature of WAHNs makes key management design an interesting research problem. To date there exist a large number of proposals trying to solve the problem of key management for WAHNs in applications with different restrictions and demands. In the following sections, we will mainly consider the following properties to evaluate key management techniques for WAHNs:

- Scalability: the total number N of nodes accommodated by the key management technique. In KPSs this parameter is typically limited by the memory available on WAHN nodes to store keying material.
- Resource Performance: computational and communication efficiency level and storage requirements on nodes.
- Feasibility for mobile ad hoc networking: ability to enable performance-aware (scalable as well as energy and time efficient) key establishment (and, thus, security) in dynamic WAHNs, where WAHN membership and size is a priori unknown.
- Connectivity properties: measures the probability that two nodes can establish a pairwise key. A key management technique with perfect connectivity enables direct pairwise key establishment.

- Security services enabled: one or a combination of confidentiality, integrity, authentication and non-repudiation.
- Robustness: resistance to security attacks (e.g. man-in-the-middle).

In PKIs, assuming all the nodes can be initiated with one (or more) valid public/private key pair(s) and a public key certificate, the scalability is unlimited. That is, any node can establish secure communications with an unlimited number of other nodes. Additionally, given the same assumptions, the connectivity probability is one. That is, any node can establish secure communications with any other node. Therefore, hereafter we will not analyze either the scalability or connectivity issues in PKIs.

2.3 Server-based Infrastructures

Server-based key management infrastructures are successfully applied in computer networks such as company LANs. To date, the most successful instance is the Kerberos infrastructure [16]. In a LAN security domain a Kerberos server keeps track of all the LAN nodes and distributes to them symmetric keys and trust information on demand. LAN nodes use these symmetric keys to further establish session keys to protect the confidentiality, integrity and authenticity of their communications. The trust information accompanying a key can also be used to verify access rights of a client node demanding access to a server residing at the LAN.

The Kerberos server can additionally be used to provide online non-repudiation services. The Kerberos server stores a log of client/server transactions by trusted clients and servers. In case of disputes, client, server and a referee need online access to the Kerberos server.

A centralized non-repudiation service cannot completely be trusted by either the client or the server, since the Kerberos server could be compromised by or be favorable to any of them. Therefore, applications demanding non-repudiation services often deploy decentralized solutions that involve just the client and the server. In addition, a decentralized solution (enabled by PKI) is more flexible than centralized solutions because client and server can apply trustworthy non-repudiation properties to their transactions without requiring online servers.

Because server-based security is not a viable (or, otherwise, optimal) option in most WAHN scenarios, there are few contributions in this research line. Perrig et al. [4] were the first authors applying the Kerberos concept in the context of WAHNs of strictly resource-constrained nodes. In SPINS [4] each node shares a secret key with a base station (BS) (here the BS takes the role of the Kerberos server). The BS resides at the WAHN with the rest of the nodes. Any two nodes use the BS to establish a common session key. This key can be applied to protect the confidentiality, integrity and/or authenticity of the communication.

Pirzada and McDonald [5] analyze the particular case where WAHN nodes have access to a network infrastructure. They propose making use of a group of (Kerberos-like) dedicated trusted servers (at the network infrastructure) for WAHN security. The trusted servers share *a priori* a pre-established secret with each end node and with each other server.

Trusted server-enabled key establishment exhibits some nice properties for WAHN of static nodes [1][2][3][30][38][39][40][41]. Firstly, since WAHN nodes are only required to store one shared key with the server, it optimizes storage efficiency in nodes. Secondly, assuming that the storage space in the server is not an issue, the size of the WAHN can indefinitely scale without affecting the storage requirements in nodes. Thirdly, since the enabled key establishment protocols uniquely rely on symmetric key cryptography, it optimizes computational performance, i.e. energy performance.

However, a centralized security solution possesses a number of weaknesses that severely impede their wide-spread application for WAHNs [1][2][3][30][38][39][40][41]. Firstly, if just the security server is compromised, the WAHN becomes insecure (i.e. the attacker reads the keys stored in the server and then use them to eavesdrop WAHN communications and/or to impersonate WAHN nodes). In typical WAHN applications the trusted server is to be left unattended in public places. This operational requirement significantly increases the risk of server compromise in respect to traditional networking applications. Additionally, WAHN nodes use wireless communications to access the security server. Consequently, an attacker succeeds in disabling the establishment of new secure communications by simply jamming the wireless link to the server.

Secondly, if the security server is down or it cannot be reached, no new secure communications can be established in the WAHN. The latter is a special issue in transient WAHNs that are sporadically and suddenly formed by WAHN mobile nodes. In some applications nodes erratically wander around a physical area. When two or more nodes come into the wireless vicinity of each other, then a WAHN is suddenly formed to exchange node information. When the nodes fall apart from each other, the WAHN progressively diminishes till it may eventually disappear. Because a large number of independent mobile nodes are deployed in the same area, more than one transient WAHN coexist in different places within that area. A centralized security approach in this context requires a security server to be present at each and every WAHN during its complete lifetime. This security system is difficult to realize because we ignore at design the location and the quantity of WAHNs.

Thirdly, centralized key distribution does not pose an energy or time efficient solution for large-scale multi-hop WAHNs. In this setting access to the server involves several hops between nodes not directly connected to the server.

A complex and time-consuming process of establishing a trusted routing infrastructure is needed to support centralized key distribution. Firstly, the routing nodes directly connected to the server establish trust. Then, the rest of routing nodes progressively establish secure links. The establishment of this infrastructure is even more complicated and energy-expensive in dynamic scenarios. Here, the secure routing infrastructure must dynamically update to changes of WAHN membership.

In a WAHN with progressively increasing membership number and density the power of the radio transceivers is automatically tuned down in order to decrease both power consumption and interference. This automatic regulation, in turn, further increases the number of hops separating distant nodes from the trusted server. Consequently, the energy and time cost to distribute a key degrades further.

The nodes in the first hops from the server are required to naïvely waste their energy to forward every petition to the server. Eventually, they either run out of batteries or they better choose not to cooperate.

In some WAHN applications nodes are configured with strictly low transmission duty cycles to save energy (and, thus maximize their longevity) or to conform regulations[7]. In such cases, intermediary nodes add significant delays to the distribution of keys.

2.4 Public Key Infrastructures (PKI)

PKIs have a tremendous deployment success in networks such as the Internet. PKI is a security architecture that provides trust for exchanging information over insecure communication channels.

In a PKI, each node has a public/private key pair. A private key is different from a public key. Although there is a mathematical unequivocal relationship between both keys, it is unfeasible to derive the private key from its sibling public key. The private key is kept private to its owner, while the public key can be distributed to other nodes.

The main component of a traditional PKI is the certification authority (CA). The CA is a globally (at least for a group nodes) trusted entity to issue and revoke public key certificates. The CA generates a public/private key pair for a node (requesting it), after verifying the identity of the node. Additionally, the CA provides the node of an authentic (i.e. signed under the CA's private key) public key certificate binding its identity to its public key. Alternatively, the node generates its own public/private key pair and, subsequently, applies for a certificate validating the authenticity of its public key.

Traditionally, the trusted CA stays online to reflect the current public/private key bindings. These bindings could change over time for a number of reasons. Firstly, a public key should be revoked if the owner node is no longer trusted or is out of the network. Secondly, a node may refresh its key pair periodically to reduce the chance of a successful brute-force attack on its private key.

PKI enables authentication, confidentiality, integrity, and non-repudiation services in a scalable and flexible way [31]. To authenticate two nodes first exchange their respective public key certificates, validate them (i.e. verify the certificate is signed by the trusted CA and the certificate has not yet expired), and finally use a public key-based authentication protocol to validate their identities. If the CA is online available, a node may additionally contact the CA to verify the validity of a communication partner certificate.

After (or concurrently with) authentication, both nodes use a public key-based key exchange protocol to establish a symmetric key (typically referred to as *session key*).

[7] For instance, the ETSI mandates a maximum 1% duty cycle in the 868.0 MHz band (which is used to allocate one of the ZigBee™ channels in Europe).

This symmetric key is used to efficiently protect the confidentiality and integrity of the rest of the conversation.

The signature of a message (or a file) with a node's private key represents a portable node fingerprint on this message (or file). Anybody in possession of the node's certified public key can validate its signature. Moreover, given that the private key is a secret exclusive to its owner, a node cannot deny having signed a message.

Certificates can be listed in Directories (another component of the PKI) to ease access to a node's public key.

To date there exist a great deal of proposals [6]-[14][16][17] for setting up PKIs in WAHNs. They can be grouped depending on the kind of CA employed:

1. Offline CA-based PKI,
2. Online partially distributed CA-based PKI,
3. Online fully distributed CA-based PKI.

In the following subsections we will review the specific operational concept of each PKI. They can all be used to provide authentication, confidentiality, integrity and non-repudiation services.

These proposals were conceived ignoring energy and time issues of WAHNs and, thus, they (with different degrees of significance) share common performance limitations for a number of reasons. Firstly, public key operations are highly CPU intensive [68]. Consequently, abusive (or repetitive) use contributes to quickly exhaust battery-powered devices. Secondly, because public keys and certificates are large (to date accepted size is 1024 bits for a wide-spread RSA key [61]) and public key operations are complex, it is unfeasible to timely (and concurrently) handle public key-based protocol transactions by nodes with strict CPU and/or RAM limitations. Thirdly, radio transmission consumes significant power [68]. Because public key-based protocol messages are large (at least one order of magnitude larger than symmetric key based protocols[8] [61]), devices waste excessive battery resources. Fourthly, because WAHNs typically support low-throughput communication channels, a public key-based exchange can significantly degrade latency.

In Chapter 3, we will present the HKMI [1], a key management concept to improve the energy and time efficiency of current PKIs for WAHNs of moderate-power devices.

2.4.1 Offline Certification Authority

A simple PKI can be enabled with just an offline CA [6][7].

[8] This means that a symmetric key-based protocol message consumes at least 10 times less energy than a public key-based protocol message. We will see in Chapter 3 that a symmetric key-based key establishment protocol, involving various messages, can improve public key-based key establishment protocol up to two orders of magnitude.

This approach provides each WAHN node with one or more public/private key pairs and digital certificates in a bootstrapping phase, i.e. before the actual WAHN exists. The CA may additionally associate multiple permissions to each public key, i.e. for what a WAHN node is trusted.

After the bootstrapping phase, when two arbitrary nodes communicate in the WAHN, they can authenticate and establish session keys using their public/private key pairs. Additionally, they can sign messages (or files).

After the bootstrapping phase, the CA is not present at the WAHN. Therefore, an attacker cannot compromise the private key of the CA by capturing it. Nonetheless, since the CA is not addressable anymore, key revocation is not possible without further WAHN self-organized control mechanisms, e.g. node reputation mechanisms.

2.4.2 Online Partially Distributed CA

The offline CA-based PKI cannot be employed in WAHN applications requiring fresh monitoring of the status of public/private key bindings. The obvious alternative is to set an online CA at the WAHN. However, a key management service based on a single CA exhibits some important problems in WAHNs. Firstly, the CA is a vulnerable point of the network, i.e. if the CA is unavailable, nodes cannot verify revocation status of a node's public key. Secondly, if the CA is compromised and leaks its private key to an adversary, the adversary can then sign any erroneous certificate using this private key to impersonate any node or to revoke any certificate.

A typical approach to improve availability of a service is replication. However, naïve replication of the CA makes the key management service even more vulnerable, since compromise of any single replica, which possesses the service private key, could lead to collapse of the entire system.

To solve these limitations, a few papers address the use of threshold cryptography to distribute CA functionalities to *s* WAHN nodes denoted servers [8]-[11].

A $(s, t+1)$ threshold cryptography scheme allows *s* parties S_1, S_2...S_S to share the ability to perform a cryptographic operation (e.g., creating a digital signature). Any *t*+1 parties can perform this operation jointly. However, a number of *t* or fewer colliding parties cannot successfully perform the same operation.

In distributed CA-based PKI, the CA private key *PK* is divided into *s* shares PK_1, PK_2...PK_S using $(s, t+1)$ threshold cryptography. Each share PK_i is assigned to a server S_i, for $i = 1,2...s$.

A number *t*+1 of partial signatures are needed in the generation of new certificates. For the CA to sign a new public key certificate, each server S_i generates a partial signature for the certificate using its private key share PK_i and submits the partial signature[9] to

[9] No extra information about *PK* is disclosed to the combiner.

one (or t+1) combiner server(s). With t+1 correct partial signatures, the combiner computes the final signature for the certificate.

In this manner, this approach increases security robustness and availability in the presence of security attacks from malicious nodes and compromised nodes. To break the key management service, now an attacker needs to expose t+1 servers.

A number of proposals [11]-[14] discuss previous partially distributed PKI models in the context of clustered WAHNs. Cluster heads play the role of server in a WAHN-wide distributed CA. Each cluster head issues, renews and revokes public key certificates to WAHN nodes within the cluster.

2.4.3 Online Fully Distributed CA

In applications with no common trusted entity, the *pretty good privacy* (PGP) [16][17] web-of-trust model allows users themselves to establish trust relationships. In this model, trust is not referred to a single or a partially distributed common CA but it is progressively built as a chain of trust relationships among all the network users.

PGP defines different trust levels (complete, marginal and no-trust) for public keys, i.e. for what and how much a node is trusted. For instance, a complete trusted party can be trusted to introduce other parties. Imagine three nodes A, B and C, where B and C completely trust A, and where B and C do not yet trust each other. After A shows trust in B to C and in C to B, then B and C trust marginally each other. After B and C satisfactorily exchange some useful data, then they may end up completely trusting each other.

Public/private key bindings and chains of digitally signed certificates can be used to realize PGP's trust model. In the previous example, to demonstrate its complete trust in node B (or C), A includes its level of trust in B (or in C) and signs B's (or C's) certificate with its private key.

Based on PGP model, Capkun et al. [15] proposed a fully self-organized PKI for WAHNs. The system allows nodes to generate their public/private key pairs and to issue certificates to other nodes. Revocation of nodes is also enabled.

This kind of key management system inherits the security vulnerabilities of fully distributed trust models. In particular, since no central trusted CA is controlling which nodes are trusted, an attacker may access the network by compromising the private key of any of the nodes. Additionally, any node may fail assessing trustworthiness of a new node and then sign a certificate for an attacker. Once misbehaving node or an attacker possesses a certificate within the web-of-trust, it is straightforward for him to fabricate and introduce new identities.

To try to reduce the risk of these attacks in a WAHN, self-organized PKIs need to be complemented with self-organized key revocation and control mechanisms, e.g. node reputation mechanisms.

2.5 Key Pre-distribution Schemes

A key pre-distribution scheme (KPS) enables nodes of a WAHN to establish keys without requiring online connection to a trusted server.

As in PKIs with offline CA, during a *bootstrapping phase* an administrator initializes the nodes with some *keying material*. The interesting (and differential, when compared to PKI) aspect of a KPS is that the keying material consists of symmetric keys (or, alternatively, information to easily generate symmetric keys [18][19]).

The computational and communication efficiency of symmetric key protocols together with the ability to enable key establishment independently of any server makes KPSs to date the only feasible option in typical wireless sensor network (WSN) applications [3][30].

However, the memory restriction of sensor nodes limits the number of keys carried by each node. This, in turn, limits the scalability and the resiliency of a KPS.

In PKI a node just needs to carry its own private key to be able to communicate with an unlimited number of users. A KPS is targeted at low-cost low-power WAHN applications where public key is prohibitive. Low-cost low-power WAHN are usually composed of inexpensive nodes without tamper resistance. In contrast to PKI, to make a KPS design scalable we let each node carry a set of symmetric keys shared with several other nodes. This leads to a side-effect problem: any exposed node that leaks its keys poses a potential compromise of communications between non-exposed nodes. The percentage of potential compromised communications of non-exposed nodes when a number N_C nodes are exposed is a critical and defining metric of a KPS, hereafter referred to as *resiliency*. For instance, a KPS distributing the same key to all the nodes has no resiliency.

Typically, a KPS enables data confidentiality and integrity services. Further enabling authentication service in a performance-aware fashion is a complex issue.

2.5.1 Generic KPS Model

All KPSs for WAHNs found in literature [20]-[30][32] can be summarized in a simple model. In this section, we discuss the generic model of key pre-distribution in the context of low-cost low-power WAHNs.

2.5.1.1 Set-Up Phase

The set-up process is typically performed by a system administrator in a secure environment distinct to the final location of the WAHN.

Generally, at this point in time the system administrator has no knowledge of the future deployment location of each individual node. Therefore, KPSs are designed to maximize the probability that any two nodes can establish a secure connection, after they are randomly deployed.

The administrator loads *keying material* (including symmetric keys – or information to easily generate symmetric keys – and optionally key and node identifiers) in each individual node using a *deterministic* or *random* distribution method.

After the set-up phase, the nodes are randomly spread in a given physical space. Nodes form one (or various) WAHNs and manage the WAHN autonomously without requiring further human intervention. The area where nodes are deployed is typically of public access. Consequently, the KPS needs to be designed to cope with node captures and key exposure.

2.5.1.2 Secure Link Formation

In most KPSs the keying material carried by nodes consists of a set of distinct symmetric keys. In KPSs based on Blom or Blundo KPS the keying material consists of encoded keying information from which a pairwise symmetric key can be easily and securely derived. In any case, hereafter we assume two nodes have found a common key out of their key sets or they have derived a common pairwise key from their encoded keying information.

Assume two nodes with a common key want to communicate. If the common key supports node authentication, to establish a secure channel both nodes firstly authenticate and then derive a session key. The session key is subsequently used to protect the communication channel. Alternatively, if the common key does not support node authentication, both nodes may either use the common key as a session key or they may derive a session key from the common key.

Nodes may need to repeatedly re-establish a secure channel because they loose connectivity.

Assuming KPS keys are only used to protect data link layer connections (such is the typical case of WSNs), in fixed scenarios a node establishes communication with a reduced number of nodes in the same wireless neighborhood. In mobile scenarios, a node may encounter and establish a secure wireless link with any other node during the node's lifetime.

2.5.1.3 Secure Communication

Nodes use the protected communication channel to exchange data.

2.5.2 Attacker Model

To assess the security level of a security system (in our case, of a KPS), formal attacker models are employed.

A widespread used attacker model [33] assumes that the attacker may eavesdrop, replay or modify any transmitted message. The attacker may also insert forged messages and impersonate one or both ends of the conversation. The attacker may additionally act as a man-in-the-middle [16].

In some WAHN applications, in which nodes are not tamper-resistant and are left unattended in public places, we need to upgrade the attacker model. Now the attacker may also physically capture nodes and read the keying material from their memory. An attacker who captures nodes *randomly* is defined as the *oblivious attacker*. An attacker who exploits information acquired in previous captures to selectively capture new nodes and, thus, progressively maximize the quantity of new tamper keying material is defined as the *smart attacker* [32].

2.5.2.1 KPS Resiliency

The resiliency of a KPS is defined as the resistance of the KPS against compromise (i.e. disclosure) of distributed keying material.

Assume N WAHN nodes carrying keying material pre-distributed by certain KPS. Assume also an attacker possessing capabilities to capture nodes and read the keying material stored in them. The *resiliency* of the KPS against node captures is measured as the percentage of communications of non-captured nodes that are compromised (i.e. deemed insecure) when a number N_c of nodes are captured [30].

For instance, assume a WAHN where all the nodes carry the same key k. Now imagine a node is captured and its copy of the key is exposed. It is easy to see that any communication that any two non-captured nodes may establish with k thereafter is not secure anymore. That is, the fraction of compromised communications is 100%.

2.5.3 System Key Pre-distribution Scheme

The simplest KPS approach is to pre-load a single symmetric key k into all nodes before deployment in the WAHN.

This approach can be applied to provide simple security in some WAHN applications. Two nodes u and v can use k (or a pairwise key k_{uv} derived from k) to encrypt their communications.

The system KPS is optimal on memory cost on nodes because each node only needs to store a symmetric key (of e.g. 128 bits). However, because every node possesses the same key k, it is not possible to confidently verify the identity of a node in the WAHN, i.e. the KPS does not support node unique authentication. The exposure of the key k in any of the nodes compromises the rest of communications.

Zhu et al. [25] proposed to use a system key k to derive pairwise keys during an initial phase of sensor node deployment in WSNs. Each node u derives a different pairwise key k_{uv} with each node v in wireless range. After this initial phase, each and every the node is supposed to erase the system key k from its memory.

Assuming that no attacker is present at the deployment area during the initial deployment phase and that no node fails to erase the system key, this system provides perfect resiliency against node captures in the post-initial phase. Otherwise, the attacker can capture a node and learn the key k, before the node erases k from its memory. Subsequently, the attacker can learn pairwise keys derived by other nodes by simply decrypting the handshake used to establish each pairwise key.

An important drawback of Zhu's et al. is that new nodes cannot be added to the WAHN, after nodes erase the system key. Consequently, it cannot be applied to WAHN applications requiring subsequent secure addition of nodes after the initial deployment (to for instance replace nodes with exhausted batteries).

Basagni et al. [27] proposed a key management system for WSNs that periodically updates a system key *k*. The WSN time is divided in regular time periods T_1, T_2 ...T_n. During each time period T_i a new key k_i is globally used to protect WAHN communications.

To make the system scalable the WSN is divided into clusters. Each cluster has its own cluster head node selected among all the nodes in the cluster. The cluster heads are to compose a network backbone for re-keying (and optionally for routing). Among all the cluster heads, one is randomly chosen in each and every time period to generate a key for the next time period. Subsequently, the new key is distributed from the key manager node to the cluster heads and ultimately to the rest of nodes.

The key k_{i+1} is generated by applying a one-way[10] hash function *h* on k_i. After receiving k_{i+1}, the key k_i shall be erased from each and every node. In contrast to Zhu et al. concept, this scheme enables secure node additions: a new node applies *i* times the hash function on the initial key *k* to obtain the key k_i for the current time period. However, as side-effect an attacker learning k_i can also easily derive all the keys for the following periods. This important security vulnerability restricts Basagni et al. concept only for WAHN applications where the attacker can eavesdrop, modify and inject messages but cannot physically capture nodes.

2.5.4 Trivial Key Pre-distribution Scheme

This KPS pre-distributes a *distinct* (and statistically independent) pairwise key to each pair of nodes of a WAHN [30][31]. For instance, assume a WAHN to be composed of three nodes *u*, *v* and *w*. The KPS assigns k_{uv} ,k_{uw} to *u*; k_{uv} ,k_{vw} to *v*; and k_{uw} ,k_{vw} to *w*.

This approach exhibits perfect resiliency. In the previous example, if *u* is captured, then the keys k_{uv} ,k_{uw} are exposed. Note that none of these keys are used to protect the communications between *v* and *w*.

The trivial KPS supports node authentication and direct key establishment in WAHNs. Because k_{uv} is a pairwise key exclusively used by the pair of nodes *u* and *v*, node *u* (or *v*) can use k_{uv} to validate the identity of a node claiming to be *v* (or *u*). Since *u* and *v* share k_{uv}, they can use this key to further derive session keys without requiring intermediaries. This quality is of special relevance in low-cost low-power WAHN designs because of its superior security strength and energy and time efficiency.

The trivial KPS is not scalable to large number of nodes, given a memory constraint *m*. Each node of an *N*-node WAHN needs to store *N*-1 keys. Therefore, the number *N* is

[10] Knowing x; y = h(x) is straightforward (and lightweight) to compute. However, knowing just y, calculating x is computationally unfeasible.

limited by the memory *m* available in nodes. For instance, a WAHN formed with nodes with memory space for 124 keys cannot get larger than 125 nodes.

2.5.5 Random Key Pre-distribution Schemes

Eschenauer and Gligor [20] proposed random key pre-distribution for WSNs to improve the resiliency of system key KPS and the scalability of a trivial KPS.

The main idea is to load, before deployment, in each sensor *u* a subset *s* of up to *m* keys (called a *key ring* and hereafter denoted as K_u) *randomly* picked from a *key pool S* with $|S|$ distinct keys. Two nodes share a key with probability $p = 1 - \frac{((|S| - s)!)^2}{(|S| - 2s)!|S|!}$ [20].

During the post-deployment phase, in order to establish a pairwise key, two sensor nodes only need to identify the common keys that they (*may*) share.

Chan et al. [21] allow two nodes to setup a pairwise key only when they share at least *c* common keys. The pairwise key of two sensors is calculated as a function of *c* keys.

In respect to Eschenauer´s KPS, Chan et al. KPS strengthens the resiliency against small-scale node captures while trading off increased vulnerability in the face of large-scale attacks. For instance, assume each node carries 200 randomly chosen keys. For *c* = 2, the amount of additional communications compromised when 50 nodes have been captured is 4.74%, as opposed to 9.52% for Eschenauer´s KPS. However, when more than 125 nodes have been compromised, the *c*-composite KPS reveals a larger fraction of communications.

The definition of small-scale is fuzzy. In the previous example, small-scale can be understood as an attack of up to 125 nodes. For *c* = 3, small-scale should be understood as lower than 80 nodes (according to Figure 2 in [21]). For greater *c*, the definition of small-scale continues shrinking.

Chan's KPS resiliency grows exponentially with the number of captured nodes, whereas Eschenauer's KPS resiliency grows linearly (refer to Figure 2 in [21]). By increasing *c*, it is harder for an attacker to compromise a significant fraction of communications with small-scale captures. However, a large-scale attack impacts much more significantly with greater *c*.

In some WAHN applications Chan's KPS may be preferred to Eschenauer's KPS because small-scale attacks are much easier to mount than large-scale attacks.

Hwang and Kim [24] indicated that allowing a small number of isolated nodes helps reducing the required number of keys in a key ring or, alternatively, improving security of a random KPS. They also proposed to use dynamic radio power adjustment to increase the degree[11] of a node when establishing keys. This technique increases the

[11] The number of nodes in wireless range of a node.

probability of finding nodes with shared keys in the same wireless neighborhood. In turn, the probability of having a fully secure connected WAHN also increases.

Di Pietro et al. [26] applied a geometric model for random key pre-distribution. For a WAHN of size *N*, Di Pietro's KPS distribute *s* keys to each node. Imagine a unit square were the nodes are randomly deployed. The number of keys in the key pool is to be calculated as $|S| \approx r^2 N / \log N + 2 \log r$, where *r* is a node's normalized wireless transmission range. They claim this approach guarantees a high probability of secure WAHN connectivity[12] even if *s* is just two [26].

One relevant drawback of the first random KPSs is that random keys cannot be used to provide unique authentication [30]. Since key rings contain keys randomly picked from the same key pool, more than a pair of nodes may use the same common key to secure their communications. Note that unique authentication is of special relevance to support access control services and to revoke misbehaving or captured nodes.

Following, a number of random KPS proposals [21][22][23][29] enable the establishment of keys that enable unique authentication. Chan et al. [21] propose to randomly pre-distribute unique pairwise keys to each pair of nodes. Du's et al. [22] solution combines Blom's scheme [18] with random key pre-distribution. Liu and Ning [23] applied a similar idea with Blundo's scheme[13] [19] (Blundo's KPS is revised in detail in the following subsection and in Chapter 4).

Similarly, Lee and Stinson [29] combine (m, r, λ, μ)-strongly regular graphs *G* with a modification on the original Blom scheme. This approach splits an *N*-node population in *m* classes with *l* nodes each. Each class is associated to a vertex of *G*. The 2*l* nodes u_1, u_2,…u_l and v_1, v_2,…v_l belonging to two adjacent classes *u* and *v* receive keys, which enable nodes u_i to directly establish keys with nodes v_i. To establish a key with nodes of the same class, nodes of class *u* need the help of a node of class *v*. Since a class *u* is adjacent to *r* distinct classes, a node in *u* can establish direct keys with $r \times l$ nodes. Then, assuming random deployment of nodes, two arbitrary nodes can *directly* establish a key with probability $p_d = (r \times l)/(N-1)$ [28]. Since non-adjacent classes *u* and *v* have μ classes $c_1, c_2, \ldots c_\mu$ in common, a node of class *u* can indirectly (through one-hop) establish a key with nodes of *v* by using any of the $\mu \times l$ nodes in $c_1, c_2, \ldots c_\mu$.

The main drawback of random KPSs is that two arbitrary nodes *directly* find a common key with a given probability *p* (generally *p* directly influences the performance of the KPS). Thus, neighboring nodes (one or more hops away) are enabled as TTP to assist establishing keys to two sensors that do not share a key in their respective key rings (this mechanism is referred as *path-key establishment* in the literature on random KPSs).

Path-key establishment is limited by the physical connectivity properties of the WAHN. The probability of finding a TTP among the neighbors directly depends on the size of

[12] A path of one or more secure links connects almost any two nodes in the WAHN.

[13] Blundo's scheme is a generalization of Blom's.

the neighborhood. A denser node neighborhood increases the probability to establish keys but this, in turn, increases interference and decreases user-throughput, i.e. the performance of the KPS and the network is traded-off [24]).

Consider now a WAHN with one or more partitions. In partitions composed of a few nodes finding a TTP among the neighbors may result often impossible.

Additionally, path-key establishment introduces two severe additional security risks in WAHNs of unattended nodes. Firstly, since each node *u* is *allowed* (and often required) to work as a TTP, it becomes a single point of compromise for *its* trusted nodes (recall this is a major vulnerability in server-based security infrastructures, which we try to overcome with KPSs) [3]. To minimize this vulnerability the random KPS needs to be complemented with an effective and efficient intrusion detection system (IDS). The IDS must provide reliable node trust and reputation metrics to reject misbehaving or compromised nodes from the WAHN. Additionally, (if possible) nodes should use more than one trusted node to establish a path key. This mechanisms however introduces further communication and computational overhead.

Secondly, a single captured node can be used to significantly reduce the resiliency of a random KPS. Assume that the nodes have been set-up with Eschenauer's KPS [20] . An attacker can capture a node, compromise its keys and drive the following path-key attack: the attacker uses different replications of the captured node at multiple WAHN locations to establish path-keys. Since path-keys are to be taken from the key ring of the trusted node [20], in each new established path-key the attacker has a new chance to learn a new key included in the initial key pool *S*.

Assume now that the nodes have been set-up with a random KPS supporting node authentication. In this case, it has been suggested by Newsome et al. [34] that a node replication attack can be "relatively simply" defended by centrally registering each identity's location. However, in WAHNs with multiple partitions the path-key attack will anyway succeed because no central entity can interconnect all the partitions. Partitions are likely to happen in mobile WAHNs and in low-cost low-power WAHNs (where a node can be in sleep mode up to a 99.90% of the time). In fully-connected WAHNs, the attacker can cause partitions and then mount the path-key attack. To cause WAHN partitions the attacker disrupts the link of some carefully selected nodes.

2.5.6 Deterministic Key Pre-distribution Schemes

The energy, connectivity and security drawbacks of using neighboring nodes as TTP are solved with deterministic KPSs that enable *direct* key sharing between any arbitrary pair of nodes. The system key and trivial key KPSs (discussed in Sections 2.5.3 and 2.5.4) belong to this classification.

Blundo's two-party [19] polynomial-based KPS provides an interesting system for WAHN security. As the trivial KPS, Blundo's KPS supports a distinct pairwise key between each and every pair of nodes of a WAHN. However, Blundo's KPS trades-off resiliency for scalability. It can be applied to WAHN applications requiring a large number of scarce-memory nodes and relaxed resiliency.

A symmetric bivariate λ-degree polynomial over a finite field F_q is a polynomial of the form $f(x,y)=\sum_{i,j=0}^{\lambda} a_{ij}x^i y^j$. Each of the coefficients a_{ij} are randomly taken from a finite field with q elements, where $a_{ij}=a_{ji}$. Note that $f(x,y)=f(y,x)$. The finite field order q is a prime number large enough to accommodate the size of a cryptographic key. That is, for a WAHN application requiring keys of 64 bits, q must be a prime number larger than 2^{64}.

A polynomial share $f(u,y)$ is calculated by evaluating $f(x,y)$ at a point u of the finite field F_q. For $\lambda \geq 1$, $f(u,y) \neq f(v,y)$, if $u \neq v$ and $u,v \in F_q$ [19].

In the set-up phase, a set-up server constructs a symmetric bivariate λ-degree polynomial $f(x,y)$. To each node u, the server pre-distributes a polynomial share $f(u,y)$. The share $f(u,y)$ is to be calculated by evaluating $f(x,y)$ at a point u of the finite field F_q. Hereafter the point u identifies (ID) the node carrying $f(u,y)$.

Imagine now that nodes are deployed in a physical area. Consider two arbitrary nodes u and v willing to establish a secure link. Node u and v carry shares $f(u,y)$ and $f(v,y)$, respectively. To be able to establish a pairwise key, both nodes firstly need to exchange their own IDs, i.e. u and v. Node u calculates a pairwise key k_{uv} by evaluating $f(u,y)$ at point v, i.e. $K_{uv}=f(u,v)$. In parallel, node v calculates a pairwise key k_{vu} by evaluating $f(v,y)$ at point v, i.e. $K_{vu}=f(v,u)$. Since $f(x,y)=f(y,x)$, then both independently calculated keys k_{uv}, k_{vu} are equal, i.e $K_{uv}=K_{vu}$.

Blundo's KPS enables node authentication. For $\lambda \geq 1$, the generated key k_{uv} is unique to the pair of nodes u and v. That is, no other node w can generate k_{uv} from $f(w,y)$.

The scalability of Blundo's KPS is independent of the memory allocated for keying material. A polynomial share $f(u,y)$ occupies $\lambda+1$ times the length of a cryptographic pairwise key. The number of nodes that can be accommodated using this KPS is $q-1$.

Blundo's KPS exhibits perfect resiliency up to $\lambda+1$ captured nodes. That is, a coalition of no more than λ nodes cannot generate pairwise keys of other non-colliding nodes.

In most KPSs reviewed in the previous sections a node carries a set of ready-to-use cryptographic keys. In Blundo's KPS each node carries encoded keying information (the polynomial share) to be processed to generate the cryptographic key. In Chapter 4 we show how to minimize the energy cost of generating a Blundo key in low-power nodes.

Key generation in Blundo's KPS incurs no substantial communication overhead. Both nodes just exchange two IDs of a few bits without involving third-party communication overhead. We will see in Chapter 4 how to further decrease the communication overhead of Blundo's KPS.

Blundo's KPS possesses interesting connectivity properties for mobile sensor networks (MSNs) [2][3]. That is, any pair of sensors can generate a pairwise key independently of the physical connectivity properties of the WAHN. However, Blundo's pairwise key

generation can be computationally expensive in sensor nodes with limited CPU capabilities because it requires λ modular multiplications and λ modular additions in F_q.

Liu et al. [23] solve this problem for nodes with low-bit CPUs without division instruction. Each sensor carries *t* distinct polynomial shares, with coefficients over $F_{q'}$, $q' << q$, $t = \log q / \log q'$. Each share is correspondingly calculated from a distinct symmetric bivariate polynomial. To generate a pairwise key a sensor concatenates the *t* partial keys of $\log q'$ bits generated from the *t* polynomial shares.

The side-effect of this solution is that now the maximum number of nodes that can be accommodated is reduced to $q'-1$. For instance, for optimal computational efficiency typical values of *q'* are $q' = 2^8 + 1$ or $q' = 2^{16} + 1$. In such cases, the scalability of the KPS is reduced to 256 or 65536 nodes. In Chapter 4 we show how to increase the scalability of the modified KPS.

S. A. Çamtepe and B. Yener [28] applied combinatorial designs to key pre-distribution. They first propose a simple KPS based on Finite Projective Planes (FPP) (refer to Chapter 4 for a complete introduction to FPP theory).

This KPS enables each and every pair of nodes of a $n^2 + n + 1$-node WAHN to find one common key within their key rings. The same key is shared by $n+1$ distinct nodes. Each node needs to store $n+1$ distinct cryptographic keys.

This KPS exhibits nice properties for WAHN security: direct key establishment, tolerance to node captures (up to *n* nodes under the smart attacker model) and no computational or communication overhead.

However, given that *n* needs to be a prime power lower or equal to *m* (the memory restriction), it is not always possible to find a design with the desired network scalability. By introducing randomness in the design, their hybrid design-based KPS augments scalability of the initial scheme to the cost of sacrificing direct connectivity.

Because a key is shared by $n+1$ distinct nodes, the KPS does not support node authentication.

To this classification belongs the DPKPS [2] presented later in Chapter 4 (see optionally [2]). The DPKPS exhibits nice properties for WAHN security, including direct key establishment, resiliency, computational and communication efficiency, network scalability and node authentication.

2.6 Summary and Conclusions

Among the existing key management options, we have identified that, to date, PKI and KPS constitute the most adequate alternatives to fundament security in WAHNs.

To date there exist a great deal of proposals for setting up PKIs in WAHNs. A PKI enables data confidentiality and integrity, data and entity authentication and non-repudiation services. However, because of the limitations of WAHN nodes, PKI enabled key establishment does not offer optimal performance for WAHNs.

Consequently, in Chapter 3 we address the energy limitations of existing PKI enabled key establishment.

In some WAHN applications, typically, in WSN applications, because of the scarce resources of the devices, PKI cannot be applied at all. We have evaluated KPS proposals enabling lightweight key establishment in WSNs. KPSs enable data confidentiality and integrity and they often enable also data and entity authentication. However, because of the restrictive demands of WSN applications, most KPSs do not exhibit adequate performance properties. Consequently, in Chapter 4 we additionally address the performance limitations of existing KPSs.

3 Hybrid Key Management Infrastructure

PKI enables confidentiality, integrity, authentication and non-repudiation services in a flexible and scalable manner [1][31][16]. In a security solution, the process of key establishment is paramount as it sets symmetric keys to further enable the services of data confidentiality, integrity and authentication.

Public key based key establishment is a limitation in WAHNs composed of moderate resource nodes. Because public key operations are highly CPU intensive and because messages of public key based protocols are large, the communication can be substantially delayed and batteries rapidly exhausted. These limitations turn to be particularly critical in scenarios where nodes need to often establish new keys with a great number of other nodes.

The HKMI [1] leverages existing PKI proposals [6]-[15] to construct a security solution offering the same security services as any PKI, i.e. confidentiality, integrity, authentication and non-repudiation. Therefore, HKMI can be applied to any WAHN application for which the underlying PKI has been conceived.

Moreover, HKMI enables key establishment with substantially improved time and energy efficiency. Our results demonstrate that with single-CA-based HKMI two nodes can establish a key up to two orders of magnitude faster (see Figure 8) and investing up to three orders of magnitude less battery resource (see Figure 6) than with single-CA-based PKI. Greater improvements are achieved when hierarchical CAs are involved.

To design the HKMI for WAHNs, we exploit the demonstrated advantages of symmetric key cryptography (SKC) over public key cryptography (PKC). First, for comparable security level the size of a symmetric key is one order of magnitude shorter than public key. For instance, till 2010 to date recommended size for widespread used RSA public keys is 1024 bits while for symmetric keys is 80 bits [61]. The length of messages exchanged by protocols based on SKC or, alternatively, PKC is proportional to the length of the keys [31]. Radio transmission (i.e. bits transmitted on the air) consumes significant power on mobile nodes [68]. Consequently, SKC-based protocols are more communication-efficient than PKC-based protocols.

Second, the operations of SKC algorithms (namely AES) are up to three orders of magnitude more computationally efficient than PKC operations (namely the popular RSA or ECC) [40][68]. Consequently, SKC-based protocols are more computationally-efficient than PKC-based protocols.

The remainder of this chapter is organized as follows. In Section 3.1, we set the scope, assumptions and objectives of this chapter. Section 3.2 provides an overall overview of HKMI and Sections 3.3, 3.4 and 3.5 and detail its components and mechanisms. The performance and the security level of the HKMI are assessed and compared with related work in Section 3.6. Finally, Section 3.7 summarizes and concludes this chapter.

3.1 Design Scope, Assumptions and Objectives

The HKMI is targeted for WAHN applications requiring security services of authentication, communication confidentiality and integrity, and non-repudiation. It is of especial interest for low-bandwidth WAHNs of static or relatively mobile and moderate-power devices.

We assume a wireless WAHN (see a simple example on Figure 1) composed of static or mobile nodes without online access to any fixed network infrastructure. Note that this assumption does not exclude nodes having access to a fixed network infrastructure sometime before joining or after quitting the WAHN.

The WAHN is formed by suddenly interconnecting a few nodes without previous planning. New nodes may sporadically join or quit the WAHN. Any node within WAHN coverage area may join. The precise composition of a WAHN is not predictable. The lifetime of the WAHN is transient. Once connected to the WAHN, each node establishes communication links with other nodes.

Typical WAHN devices, considered in this chapter, are PDAs, mobile phones, and embedded systems in portable devices. These devices have moderate computing power and storage resources as well as limited battery power life. We assume nodes possess similar resource capabilities. The nodes are capable of computing intensive public key operations to the cost of allowing sensitive communication delays and draining of batteries. For instance, an RSA signature takes 86 milliseconds on a 206-MHz PDA (see Table 2).

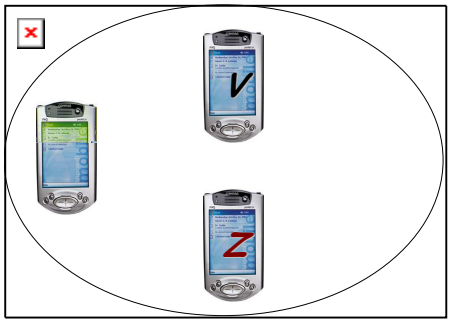


Figure 1. A WAHN formed of four PDAs

We target civilian applications in which devices are carried/wore or placed around human users, i.e. nodes are not generally left unattended. Therefore, the risk of node compromise by an attacker is low. We also assume honest and cooperative nodes.

An outside attacker may exploit the vulnerabilities of wireless transmissions to anonymously eavesdrop, modify, replay or inject bogus messages.

For the descriptions in the rest of the chapter, we assume a WAHN with P nodes. We use u, b, w, v, y, x and z to refer to some generic nodes of the WAHN.

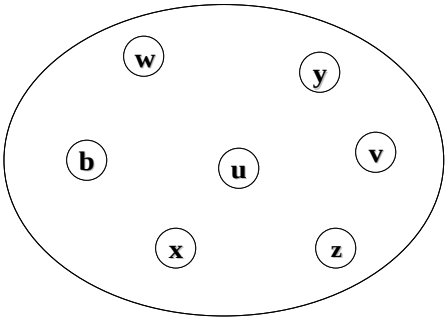


Figure 2. WAHN used to explain the HKMI

Table 1 summarizes the notation used throughout this chapter.

Table 1 Summary of notation in this chapter.

b ,u, v, w, x, y, z	Generic nodes of the WAHN
ID_X	Unique identifier associated to a generic node *x*
TP	Trusted Portal node
TP-X	Generic node *x* serving as Trusted Portal
D_{TPX}	Domain containing the trusted nodes of *TP-X*
K	Secret key
K{m}	Symmetric key encryption of message *m* with *K*
$S_{Y,TPX}$	*TP-shared-key*: Long-term key shared by *y* and *TP-X*
F()	Collision-resistant one-way pseudorandom function
$K_{V,Z}$	Key shared by generic nodes *v* and *z*
K_{TPXdel}	Delegation Key: Key issued by *TP-X* (for delegation)

3.2 HKMI Overview

In this section we briefly overview the HKMI and compare with related work.

3.2.1 Concept

The HKMI is a security infrastructure providing efficient confidentiality, integrity, and authentication and non-repudiation services in WAHNs.

Public key cryptography is required to provide secure non-repudiation services. Public key cryptography can also be used to, by means of a PKC-based key establishment protocol, derive symmetric keys for confidentiality, integrity, and authentication.

HKMI relaxes time and energy overhead of PKC-based key establishment by providing instead an ad hoc symmetric key distribution architecture at the WAHN.

The ad hoc symmetric key distribution architecture consists of two components: the Trusted Portals (TP) and the TP domains (see Section 3.4). A TP domain contains a set of nodes directly trusted by a TP. The TP generates and distributes symmetric keys to the nodes in its domain. A WAHN can contain one or more TPs. TPs also cooperate to distribute keys to nodes in different TP domains.

The operations of the HKMI to enable the symmetric key distribution architecture can be divided in three fundamental phases:

1. Initialization of nodes with PKI (see Section 3.3),
2. Trusted portal establishment (see Section 3.4),
3. Nodes trust and key establishment (see Section 3.5).

During phase (1.), WAHN nodes get a digital certificate from a WAHN PKI [6]-[15]. The digital certificate contains trust information and the public key of the owner node.

In (2.), each node independently selects another member of the WAHN and then establishes a strong trust relationship.

The selected member could be chosen by different strategies: randomly or deterministically. If we assumed that one or a set of powerful and secure nodes are always available at the WAHN, then these nodes can be deterministically selected by other less powerful nodes in joining the WAHN (this is a trivial case out of the scope of this thesis).

We assume we ignore which specific nodes will be part of the WAHN before it is formed. We also assume nodes with similar resource capabilities. Thus, we adopt random selection of nodes. This option increases security strength and availability and distributes the computational charge evenly to a set of selected nodes. Optionally, a minimal trust level (TL) can be required on selected nodes.

Finally in step (3.), in addressing other nodes of the WAHN, any node *y* uses its selected node *x* as an intermediary, e.g. *y* uses *x* to establish a key K_{YW} with another node *w* or, even, to get trust information TL_W related to *w*. Thus, hereafter we call nodes with same role as *x Trusted Portals* (TP). Intuitively, TPs constitute *ad hoc established and distributed* TTPs to securely access other nodes of the WAHN.

3.2.2 Comparison with Related Work

Our work completes previous WAHN PKI proposals [6]-[15] by adding an ad hoc symmetric key distribution architecture. It improves the time and energy performance of PKI-enabled key establishment protocols as demonstrated in Section 3.6.

Moreover, our work compares advantageously with Pirzada and McDonald [5] proposal. They propose making use of a group of dedicated trusted servers, which reside at a network fixed infrastructure, to enable WAHN security. The trusted servers share *a priori* a pre-established secret with each end node and with each other server.

Pirzada and McDonald approach exhibits critical vulnerabilities for WAHN security. Firstly, if any trusted server is compromised, the WAHN becomes insecure (i.e. the attacker reads the keys stored in the server and then use them to eavesdrop WAHN communications and/or to impersonate any WAHN nodes). Secondly, WAHN nodes use wireless communications to access the security server. Consequently, an attacker succeeds in disabling the establishment of new secure communications by simply jamming the wireless link to the server. Thirdly, if the WAHN looses connection to the server, no new security associations can succeed.

In our work nodes of the WAHN constitute potential trusted servers, i.e. the TPs. TPs are established dynamically on demand. Therefore, the TP *does not* share any pre-

established secret with each end node and with each other TP a priori. Instead, the secrets are established on demand, each time a new node sets trust with a TP.

In HKMI a TP just stores keys of nodes in its TP domain. Therefore, by compromising a TP the attacker may just eavesdrop communications from or impersonate these nodes.

HKMI relies on a PKI. Therefore, if a TP is unavailable (either as a result of an attack or a failure), its trusted nodes can still establish security associations with other nodes of the WAHN. Moreover, each of these nodes can establish a new own selected TP.

3.3 Initialization of Nodes with PKI

A PKI for WAHNs [6]- [15] underlies the HKMI. The PKI can operate either with an offline CA [6][7], with an online partially distributed CA [8]-[14] or with an online fully distributed CA [15] (see corresponding thorough reviews in Section 2.4).

In HKMI with offline CA, each node *x* obtains its public/private key pair and public key certificate sometime before joining the WAHN [6][7]. In HKMI with online CA, nodes get their public key certificates and keys in joining the WAHN [8]- [15] (see Figure 3).

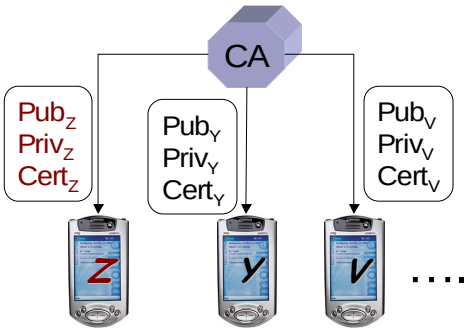


Figure 3. Initialization of Nodes in WAHN PKI

The certificate of node *x* digitally binds its identity ID_X with the corresponding public key. The certificate can additionally include the level of trust TL_X in the public key of node *x*.

Furthermore, when operating with an online CA, other operations of the PKI such as certificate renewal and revocation can be enabled.

The PKI is to be selected depending on the target application. A PKI operating with either an offline CA or an online partially distributed CA can be applied to WAHN applications where a single administration owns and controls the WAHN devices, e.g for medical WAHNs enabling patient care and monitoring. A PKI operating with an online fully distributed CA applies to WAHN applications with multiple independent administration entities, e.g. for automotive WAHNs enabling car-to-car communication.

In the remainder of this chapter we assume any two nodes of the WAHN can validate the authenticity of each other's public key certificate.

3.4 Trusted Portal Establishment

In joining the WAHN, each node *v* arbitrarily selects another node *y* from the ones present at the WAHN. Then, both nodes mutually authenticate by using their certified public/private key pairs. This mutual authentication establishes a bi-directional trust relationship between both nodes, which is required for the process of "Nodes Trust and Key Establishment" (see Section 3.5).

Assume that, from a WAHN with nodes *u*, *b*, *v*, *w*, *x* and *z*, node *y* selects node *x* as its initial trusted node. In future communications, *y* will use *x* as a portal to address other nodes of the same WAHN securely and efficiently. Therefore, we call *x* a TP for *y*. In the following we use *TP-X* to denote "*node x serving as TP"*.

A node *y*, whose current TP is *TP-X*, must establish a new TP, when node *x* quits the WAHN.

In a simple HKMI all the WAHN nodes have the same level of trust associated to their public keys, i.e. trust or no trust. In this case, all the nodes possessing a valid certificate can serve as TPs to other nodes in the WAHN.

In a more complex HKMI only nodes with special permissions are allowed to serve as TPs in the WAHN. For instance, using PGP's terminology [17], node *x* can act as TP if and only if its public key is associated a *complete* trust level. The public key of one such a node is associated a complete trust level at "Initialization of Nodes with PKI ".

3.4.1 Trusted Portal Domain

Because each and every node of a WAHN must follow the "TP Establishment" process and TPs are randomly selected, more than one node may establish initial trust with the same node *x*.

We define as *TP-X domain* the group of WAHN nodes associated to *TP-X*. From now on, we use D_{TPX} to denote *TP-X domain*. For instance, in Figure 4, domain D_{TPX} includes *y* and *w* (note that *y* and *w* are depicted as *TP-Y* and TP-W because they also have respective TP domains) as trusted nodes of *TP-X*.

TP-X is the domain administrator of its own domain D_{TPX}, i.e. *TP-X* decides when to accept a node in D_{TPX} or when trust relationships with one or more trusted nodes of D_{TPX} expire.

To accept a node *y* in its domain, a TP must first authenticate the identity of *y*. As any other node in the WAHN, the TP also wants to save energy by using the HKMI. Accordingly, a TP rejects TP Establishment requests from new nodes if it already invested too much energy in establishing strong trust relationships (see analysis in section 3.6.3).

3.4.2 Generation of WAHN Trust Graph

A consequence of the "TP Establishment" process is that one or more TPs are set in the WAHN. Because these TP nodes are selected randomly, a random trust graph connecting different nodes in the WAHN is generated.

We can guarantee the continuous existence of a random trust graph without isolated cycles under the following two conditions:

1. *Each and every node of the WAHN must dynamically initialize trust with an own selected TP*, i.e. a node repeats the "TP Establishment" process in joining the WAHN and when its TP disappears.
2. *A node, which is serving as TP in the moment it selects its own TP, must choose as TP a node not included in its TP domain or sub-domains* (e.g. in Figure 4, *TP-Y* cannot select nodes *v* or *z* as TP). If this condition cannot be satisfied for a node (e.g. *x* in Figure 4), then such node should not select any TP.

For example, for a WAHN with nodes *u*, *b*, *w*, *v*, *y*, *x* and *z*, the "TP Establishment" process may happen as follows. Firstly, nodes *v* and *z* happen to establish trust with *y*, which then serves as *TP-Y*. Secondly, node *y* happens to establish trust with *x*, which then serves as *TP-X*. Thirdly, nodes *u* and *b* happen to establish trust with *w*, which then serves as *TP-W*. Fourthly, node *w* happens to also select *x* as TP. Finally, node *x* does not choose any other node. This instance has generated a random trust graph (see Figure 4) where one or more intermediary TPs interconnect any pair of nodes.

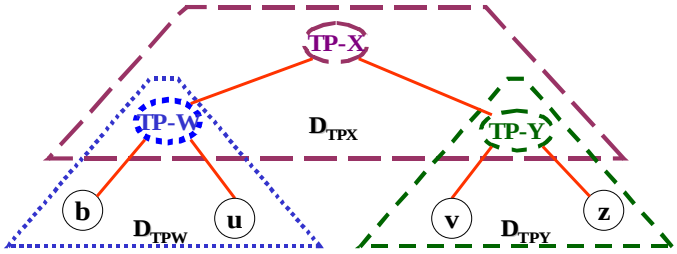


Figure 4. An example of a random trust graph generated by "TP Establishment" (the **straight** lines connecting nodes denote strong trust relationships)

The previous algorithm exhibits two potential limitations. First, it may generate a trust graph of excessive depth. Second, it generates a tree trust graph (similar to that depicted on Figure 4) in which a TP becomes the root (e.g. *TP-X* on Figure 4). Because the root is a key element of the graph, by disabling the root the attacker disrupts the HKMI. To solve these two limitations, some arbitrarily selected nodes can generate redundant paths and short-cuts in the graph.

At the end of the "TP Establishment" process two arbitrary TPs are interconnected by either a direct trust relationship or by a set of indirect ones. Additionally, a trusted path of TPs interconnects two arbitrary nodes in different TP domains.

3.4.3 Trust Path Route

A simple way to discover the shortest trust path of TPs connecting two end nodes is to let each TP node to internally store a dynamic trust routing table, *TP-Route*[14], which indexes the IDs of target nodes and their respective TPs. The TP-Route table also enables a TP to know which nodes belong to its TP sub-domains, information of special relevance in the "TP Establishment" process.

In the event of a WAHN membership change, which in turn implies that a node joins or quits a TP domain, all the TP-Route tables need to be updated. For that, a punctual notification, including the IDs of the joining/quitting node and of its TP, is triggered from the affected TP node to its parent TP and children TPs. This notification is to be forwarded in the same way to the rest of TPs.

All these messages can be cryptographically protected to guarantee the confidentiality and integrity of the process as well as the anonymity of TPs to passive eavesdroppers.

3.4.4 Trust Initialization Protocol

In the moment of establishing a TP, a node *y* may also have its own TP domain *TP-Y*, i.e. some nodes *v* and *z* may have already established *y* as their TP. Thus, before issuing a TP service request to an arbitrarily selected node *x*, by consulting its TP-Route table *y* checks that *x* is not included in *TP-Y* domain or sub-domains. Then, assuming *x* is finally chosen, the following protocol enables *y* to establish *x* as its TP:

$$\text{Y}\rightarrow\text{X}: \textit{TP\_Service\_Request} \quad (1)$$

$$\text{TP-X}\leftrightarrow\text{Y}: \textit{PKC challenge-response authentication} \quad (2)$$

$$\text{TP-X}\rightarrow\text{Y}: \textit{TP\_Service\_Accept}, K\{S_{Y,TPX}, T\} \quad (3)$$

In message (1), *y* requests a TP service to node *x*.

In (2), assuming that node *x* is cooperative, *x* and *y* mutually authenticate using certified public keys and agree in a session key *K*.

In (3), *TP-X* sends to *y* a long-term shared symmetric key $S_{Y,TPX}$ encrypted and integrity-protected with *K* and a timestamp *T*. Finally, *y* sets *TP-X* as its TP and, similarly, *X* sets *Y* as one of its trusted nodes.

In the rest of the chapter, we will use the term *TP-shared-key* and the notation $S_{node,TPnode}$ to refer to a long-term symmetric key shared between a node and its TP or to any of the keys k_i derived from it, interchangeably. For instance, in the protocol above $S_{Y,TPX}$ is a long-term symmetric key shared by *y* and *TP-X*. Note that this key enables *y* and *TP-X* to authenticate and protect future communications without the need to further run PKC-based authentication protocols.

[14] The TP-Route table does not typically match any table to discover network routes. Not to be confused.

3.5 Nodes Trust and Key Establishment

TPs can be used as *ad hoc distributed* TTPs to distribute keys and related trust information within the WAHN.

The first use case is when two arbitrary nodes *v* and *z* of the same TP domain want to establish a shared key K_{VZ}. In such a case, their TP, e.g. *TP-Y* on Figure 4, acts as a TTP providing them of the shared key K_{VZ}. Similarly, a common TP can vouch for nodes in its TP domain. For instance, *TP-Y* can associate *v*'s identity to the symmetric key K_{VZ} distributed to *z* and *z*'s identity to the symmetric key K_{VZ} distributed to *v*.

In simple trust models K_{VZ}, ID_V and ID_Z are sufficient to enable key establishment and mutual authentication of nodes *v* and *z*. In web-of-trust models, *TP-Y* additionally includes *recommendation values*, which enable both nodes *v* and *z* to respectively evaluate the level of trust that *TP-Y* has on their communication partner.

The second use case is when two arbitrary nodes *v* and *w* of different TP domains want to establish a shared key K_{VW}. The previous model with a single TP can be easily extended to several TP domains. In this case, various TPs need to cooperate to securely distribute shared keys and/or vouch for nodes in different TP domains. For instance, *TP-Y* delegates to its parent TP, i.e. *TP-X*, to vouch and distribute keys in D_{TPX} related to *TP-Y* trusted nodes.

The process of "Nodes Trust and Key Establishment" requires trust relationships established during "TP Establishment" to be bi-directional. First, *z* or *v* accept vouching and/or a key from *TP-Y* if and only if they trust *TP-Y*. Second, *TP-Y* only vouches for and distributes keys to its trusted nodes.

3.5.1 Trust and Key Distribution Protocol

In this section we describe the Trust and Key Distribution (TKD) protocol, a protocol to distribute trust and keys across TP domains. For simplicity's sake we assume below a simple underlying PKI trust model. Assuming that *v* and *w* on Figure 5 want to establish a common key, the TKD works as follows (see also Figure 5):

$$\text{V}\rightarrow\text{TP-Y: } S_{V.TPY}\{KeyReq\ (ID_W, ID_V), T_1\} \quad (1)$$

$$\text{TP-Y}\rightarrow\text{TP-X: } S_{Y.TPX}\{KeyReq\ (ID_W, ID_V), \quad (2)$$

$$\text{TP-X}\rightarrow\text{W: } S_{W.TPX}\{K_{V.W}, ID_V, T_3\}, ticket_V \quad (3)$$

$$\text{W}\rightarrow\text{V: } ticket_V \quad (4)$$

In step (1), node *v* requests *TP-Y* a key for *w*. This message is encrypted under $S_{V,TPY}$ to guarantee the confidentiality of the process as well as the anonymity of *TP-Y* and of the involved nodes *v* and *w* against eavesdroppers.

To protect against message replay and modification attacks, the messages must be additionally integrity protected, e.g. by including message authentication codes (MAC) as well as timestamps T_1, T_2 and T_3.

In (2), *TP-Y* decrypts message (1) and obtains the included timestamp T_1. *TP-Y* computes a *Delegation Key* K_{TPYdel} by applying a pseudorandom function $F(.)$ with $S_{V,TPY}$ and T_1 as inputs, i.e. $K_{TPYdel}=F(S_{V,TPY}, T_1)$.

With K_{TPYdel}, *TP-Y* delegates to other TPs to vouch for *v* and distribute keys associated to *v*'s identity in their domains. Note that the *Delegation Key* also enables other TPs to communicate securely with *v* (see further steps below).

TP-Y constructs message (2) by including K_{TPYdel} and *v*'s key request. It then encrypts message (2) using the *TP-shared-key* with *TP-X* (the next TP in the trust path), i.e. $S_{Y,TPX}\{KeyReq(ID_W, ID_V), K_{TPYdel}, T_2\}$.

Finally, *TP-Y* sends to *TP-X* message (2). In this manner, *v*'s key request is forwarded to a TP in a different domain.

In (3), decryption of message (2) with $S_{Y,TPX}$ transmits to *TP-X* (*w*'s TP) *TP-Y*'s trust in ID_V. *TP-X* randomly generates a new shared key $K_{V,W}$ for *v* and *w*. *TP-X* encrypts $K_{V,W}$ and ID_V using its *TP-shared-key* with *w*, i.e. $S_{W,TPY}\{K_{V,W}, ID_V, T_3\}$. *TP-X* also creates a *ticket* for *v* secured with K_{TPYdel} containing $K_{V,W}$ and ID_W, i.e. $ticket_V = K_{TPYdel}\{K_{V,W}, ID_W, T_3\}$. *TP-X* sends to *w* message (3).

In (4), node *w* forwards $ticket_V$ to *v*. Finally, *w* obtains $K_{V,W}$ by decrypting message (3) with $S_{W,TPX}$. In parallel, *v* obtains $K_{V,W}$ by decrypting message (4) with K_{TPYdel}.

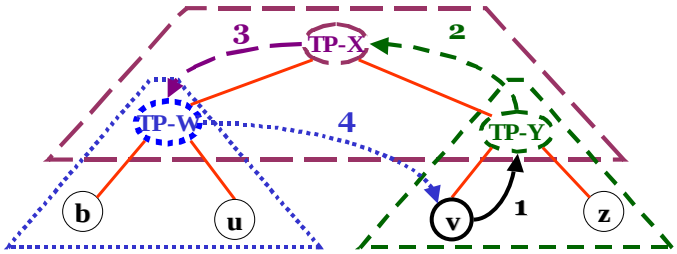


Figure 5. Details of the TKD protocol

In PKI web-of-trust models, the messages of the TKD protocol additionally include *recommendation values* $R_{target}^{voucher}$ on the identities of the participant TPs and end nodes. For instance, in the TKD protocol above, *TP-Y* includes in message (2) a recommendation value R_V^Y on ID_V for *w*, which is forwarded to *w* in message (3), together with a recommendation value R_Y^X on ID_Y issued by *TP-X*. Observe that, on this basis, *w* can derive its own trust level TL_V on *v* [62][17] (A detailed description of trust evaluation algorithms is out of scope of this thesis).

Additionally, *TP-Y* includes in message (2) an encrypted recommendation value $S_{V,TPY}\{R_X^Y\}$ on ID_X for *v*, which can be forwarded to *v* in message (4) together with an encrypted recommendation $K_{TPYdel}\{R_W^X\}$ value on ID_W issued by *TP-X*. Observe that, on this basis, *v* can derive its own trust level on *w*.

3.6 Analytical Performance Evaluation and Comparison to Previous Work

In this section we analytically study the performance efficiency of the HKMI and demonstrate its improved performance for WAHN applications by comparing with PKI. We also discuss the security level of HKMI for different WAHN applications.

We compare the cost of protocols to establish keys enabled by HKMI and PKI. Once a symmetric key is established, it can be used for node and/or data authentication and/or for data confidentiality and integrity protection.

To avoid impersonation or man-in-the-middle attacks, two arbitrary nodes *v* and *w*, which want to establish a key K_{VW}, need also to assess the authenticity of the node they are establishing the key with [31]. This can be achieved in PKIs by using an X.509 strong two-way authentication protocol with key establishment (a similar protocol is included within the SSL/TLS protocol suite). In HKMI, given that nodes *v* and *w* trust their respective TPs, the TKD protocol can be used to establish the key K_{VW}.

For simplicity's sake, in the following sections we assume that every WAHN node holds a public key certificate signed by a *common* CA and the corresponding CA public/private key pair. Because of this simplification, note that our results on computational and communication overhead are a best-case for X.509. That is, the overhead incurred by X.509 would be even worse if we considered a chain of certificates to be exchanged and validated till finding a certificate signed by a common CA.

For evaluating HKMI, we further assume a WAHN with *P* nodes, from which *N* act as TPs. We use N_{AV} to denote the average number of intermediary TPs in the shortest trust path between any pair of WAHN nodes.

Because of the popularity, acceptance and wide-deployment of RSA and AES in current products we tested RSA and AES. To date most mobile devices are capable of working both in ad-hoc and infrastructure modes (such is the case of Bluetooth-enabled mobile phones and WiFi- and Bluetooth-enabled PDAs). RSA is employed in widely-deployed security suites such as SSL/TLS for infrastructure networks. Most devices to date include RSA public keys that can be used interchangeably for security in either of both modes.

3.6.1 Communication Cost

In this section we compare the communication cost of establishing a key with TKD against X.509.

We count and compare bytes sent on the network by each protocol. We assume a single-hop WAHN because it simplifies the analysis. Considering a multi-hop network complicates the derivation of formulae, which need to consider messages transversing multiple hops. Alternatively, an average of hops between two end nodes can be computed to derive simple formulae. However, since formulae need to finally be compared, the effect of multiple hops disappears.

Consider a small or medium sized WAHN where WAHN nodes are in direct wireless range of each other (this is the case of most ZigBee-enabled WAHNs). That is, two arbitrary WAHN nodes (this includes two contiguous TPs of the trusted path) are separated by a single-hop, i.e. a message sent from *v* to *w* travels the WAHN just once.

We also consider typical sizes of RSA public keys and certificates as well as AES symmetric keys and encrypted messages [16]. However, the results in this section are implementation independent, i.e. we do not consider the message formatting imposed by different communication or security standards. Likewise, any overhead added by headers of the lower layers supporting the security messages is also ignored or by practical networking issues (such as retransmissions or message fragmentation).

Note however that a practical implementation further increases the bandwidth usage of public key-based protocols. For instance, public key certificate of sizes typically exceeding the 256 bytes occupies at least three ZigBee network messages. In such a case, ZigBee adds from 17 up to 53 bytes per message.

Let us compare the TKD and the X.509 protocols. During the X.509 protocol two arbitrary nodes *v* and *w* exchange two public key certificates, four timestamps, two node identifiers, two private key signed messages and two public key encrypted symmetric keys [31]. Then,

$$BWCost^{X.509} = 2 \times (Cert + 2 \times T_i + ID + Sign + RSAEnc)$$

During the TKD protocol, N_{AV}+3 symmetric key encrypted messages are exchanged by nodes *v* and *w* and by the N_{AV} intermediate TP nodes:

$$BWCost^{TKD} = (N_{AV} + 3) \times TKDMessage$$

Let us use N_{BWEQ} to denote the average number N_{AV} of TPs for which $BWCost^{X.509} = BWCost^{TKD}$. Then, for $N_{AV} < N_{BWEQ}$, HKMI-enabled trust and key establishment outperforms PKI. For instance, in a WAHN application using public key certificates of 256 bytes (just including a public key and a digital signature of 1024 bits), timestamps of 8 bytes, symmetric keys of 128 bits and the cipher AES-128, N_{AV} should be lower than. $N_{BWEQ} = 32$. Note that establishing a key with TKD can improve the communication overhead of X.509 up to three orders of magnitude (see Figure 6).

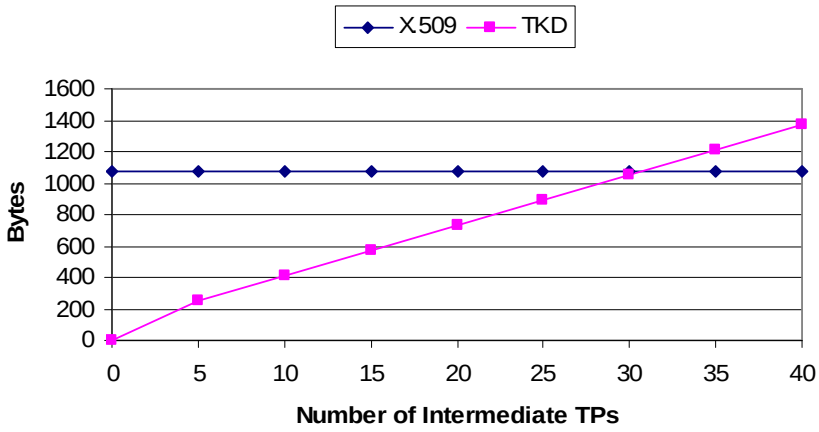


Figure 6. Communication cost of key establishment

The number N_{BWEQ} can be used as an additional parameter to control the maximum number N of TPs in a WAHN with P nodes, such that $N_{AV} \leq N_{BWEQ}$.

Let us derive the value N for a worst-case trust connectivity scenario. The worst-case scenario exhibits the longest possible average path length between any pair of nodes. This case leads to maximum communication cost because TKD packets transverse the maximum number of intermediary TPs.

Doyle and Graver [67] proved that a simple path graph exhibits the longest possible average path length $L_{AV} = (n+1)/3$ of any graph with n vertices and n-1 edges. Therefore, in the worst-case scenario the N TPs are subsequently disposed on a simple trust path (see an instance on Figure 7). Furthermore, a packet between any pair of nodes transverses at most $N_{AV} = (N+1)/3$ TP nodes. Therefore, $N = 3N_{AV} - 1$ and, finally,

$$N \leq 3N_{BWEQ} - 1 \text{ (worst-case)}$$

The formula above sets an upper bound in the number N of TPs that shall be part of the WAHN, for the worst-case scenario. In the worst-case, for $N < 3N_{BWEQ} - 1$ the communication efficiency of the hybrid approach outperforms PKI. For instance, $N < 95, N_{BWEQ} = 32$. Naturally, in other scenarios, where the N TPs are not disposed along a simple trust path, the maximum number N of TPs allowed is even greater.

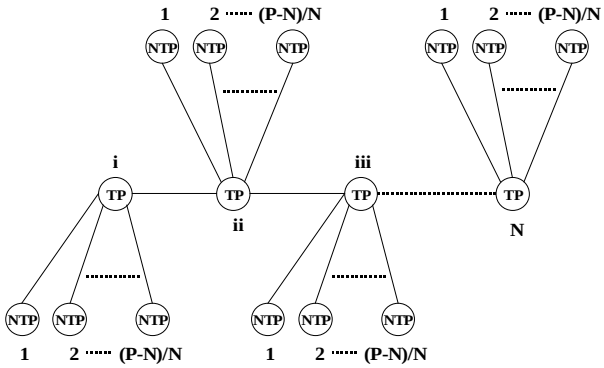


Figure 7. Worst-case trust graph scenario

3.6.2 Computational Cost

In this section, we compare the TKD and X.509 protocols in terms of computational cost.

The X.509 protocol requires two nodes *v* and *w* to compute four signature verifications, two signature generations, two public key encryptions and two private key decryptions [31]. Then:

$$CCost^{X.509} = 6 \times RSASigVer + 4 \times RSASigGen$$

The TKD protocol requires both nodes *v* and *w* and the N_{AV} intermediate TP nodes altogether to compute N_{AV}+2 symmetric key encryptions and N_{AV}+2 symmetric key decryptions. Then:

$$CCost^{TKD} = (N_{AV} + 2) \times AESEnc + (N_{AV} + 2) \times AESDec$$

We have developed a testing environment using Microsoft® CryptoAPI 1.0 [63] and Szymon Stefanek's AES C++ Class [64] on an iPaq Pocket PC with ARM SA1110 CPU at 206 MHz. Table 2 presents a summary of the cost (measured in milliseconds) to compute typical RSA public key and AES symmetric key operations. We consider RSA keys of 1024 bits and AES keys of 128 bits.

Table 2. Computational effort (milliseconds)

Operation	Settings	Timings (ms)
RSA Signature Generation/Decryption	Modulus 1024-bit; Public exponent 3	*83*
RSA Signature Verification/Encryption		*3.90*
AES Encryption	Plaintext Length 256 bits; Key Length 128 bits	*0.0235*
AES Decryption	Ciphertext Length 256 bits; Key Length 128 bits	*0.0224*

Figure 8 depicts $CCost^{X.509}$ and $CCost^{TKD}$ with values of Table 2, for different number N_{AV} of TP nodes. Let us use N_{CEQ} to denote the number of TPs for which

$CCost^{X.509} = CCost^{TKD}$. As Figure 8 shows the TKD protocol outperforms the X.509 for a number N_{AV} of TP nodes lower than N_{CEQ} = 7700 TPs. Note that establishing a key with TKD can improve the computational overhead of X.509 up to two orders of magnitude (see Figure 8).

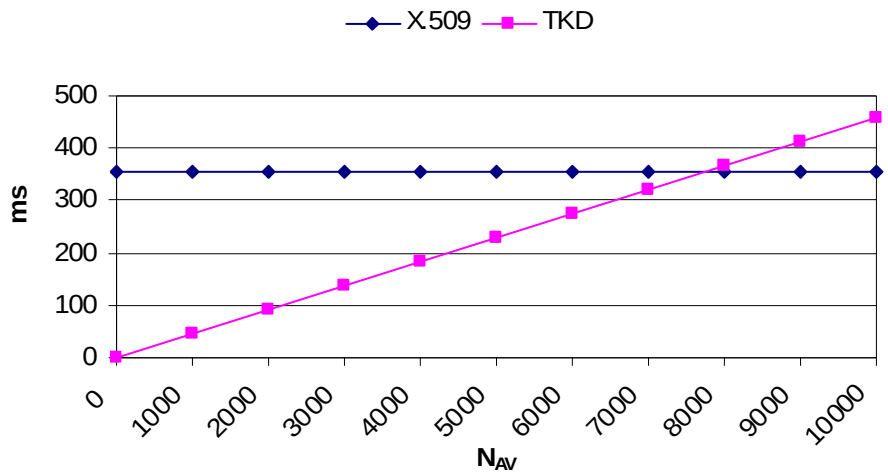


Figure 8. Computational cost of key establishment (measured in milliseconds)

Recall that the worst-case scenario (see Figure 7) fixes $N_{AV} \leq 32$ for improved communication efficiency. In such case, the computational overhead of establishing a key is negligible.

3.6.3 Analysis of a Mobile Scenario

In this section, we analyze the overhead introduced by HKMI in mobile scenarios where the WAHN membership changes dynamically.

The results in sections 3.6.1 and 3.6.2 are valid for static scenarios where each and every node only establishes one TP. Once a set of TPs is established in the WAHN, all the nodes can establish keys by using the TKD protocol.

In mobile scenarios, it is likely that nodes sporadically join and quit the WAHN (see Figure 9). If a quitting node is a TP (e.g. *y* on Figure 4 and Figure 9), then the nodes of that TP-domain must re-establish an own TP. Additionally, in node joining/ quitting events TP-route messages are flooded in the WAHN.

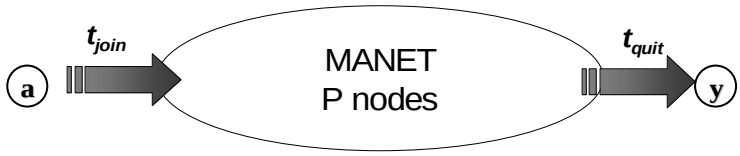


Figure 9. WAHN mobile scenario

To evaluate the performance of establishing a key with HKMI in dynamic WAHNs we have to consider the costs of the maintenance of the TP-Route table and of the "TP Establishment" process, in addition to the performance of the TKD protocol in Sections 3.6.1 and 3.6.2.

Firstly, let us analyze the communication overhead $BWCost^{TP\text{-}Route}$ introduced by the TP-Route table update process. Each TP-Route notification message goes from the issuing TP to parent and children TPs. Likewise, the notification is forwarded to the rest of TPs in the WAHN. Then[15]:

$$BWCost^{TP-Route} = (N-1) \times Notification$$

Let us now analyze the maximum number of TP nodes allowed in order to consider $BWCost^{TP\text{-}Route}$ to be significant in low-rate WAHNs. We consider $BWCost^{TP\text{-}Route}$ to be significant when it goes above 1% of the available WAHN bandwidth.

Let us assume a WAHN membership change rate of *one* node each *t* seconds and consider a 70% network throughput [66] (i.e. 30 % lower layers overhead).

Each TP-Route notification message is the concatenation of two node IDs encrypted under a symmetric key. For instance, in WAHN applications using node IDs of 8 bytes (which enables WAHNs of up to 2^{64} nodes) and AES-128 bit encryption, the notification message takes just 16 bytes. Because of the lower layers overhead the message occupies 20.8 bytes.

Bluetooth® offers a bit rate of 1 Mbps and a single-hop Bluetooth® WAHN contains at most 8 active nodes. Therefore, because $BWCost^{TP\text{-}Route}$ for $N = 8$ is extremely low (see Figure 10), TP-Route notifications incur no significant overhead on Bluetooth WAHNs (see Figure 10).

In the 2.4 GHz band ZigBee™ offers a bit rate of 250 kbps. The theoretical maximum number of devices of a ZigBee network using 64-bit addressing is 2^{64}. However, in practice a single-hop ZigBee network should be composed of just a few hundred nodes to avoid congestion, interferences or delays. For instance, imagine a network where each node needs to send a data packet of 250 bits each second. At 250 kbps each packet takes 1 millisecond. Assuming strictly fair and ordered medium access and ignoring other sources of overhead (such as medium access delay, processing delays, or network management traffic), the maximum number of devices in this scenario is just 1000 nodes. If the data packet increases its size by one factor, then the maximum number of nodes is just 100.

In a ZigBee WAHN, if the membership change rate is slower or equal than $t = 10$ seconds, the maximum number N of TPs can be up to 300 (see Figure 10). A number of up to 300 TP nodes results high enough to serve a network bounded in practice to a few hundred nodes.

[15] Note that (although ignored in this analysis) in single-hop WAHN scenarios a much more efficient protocol could alternatively *broadcast* the TP-Route notification message without the need for N-1 retransmissions, thus, substantially reducing communication overhead.

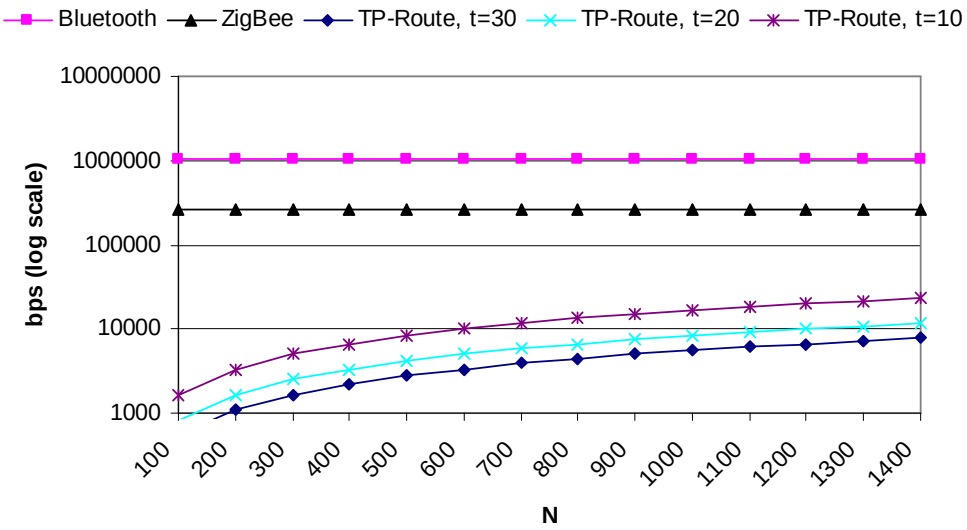


Figure 10. TP-Route update communication cost. The figure compares the available Bluetooth and ZigBee data rate against the required $BWCost^{TP\text{-}Route}$

The computational overhead added by the TP-Route update process can be neglected. Considering a WAHN with 95 TP nodes (recall that N = 95 for comparable key establishment communication overhead in the worst-case trust graph scenario of Figure 7), then a TP-Route notification message is both encrypted and decrypted 95 times. Assume an average WAHN membership change rate of t = 10 seconds and AES encrypted/decrypted messages of 128 bits. Then, the computational cost results in 2.18 milliseconds, shared by 95 TPs (2.18/95 seconds/TP) each 10 seconds.

Now let us analyze the computational effort $CCost^{TPEst\text{-}V}$ of the "TP Establishment" process on a given node v for an interval of time T_V. During T_V, node v needs to establish trust every time its TP node quits and every time it accepts TP service requests from other nodes. Let us use q_V to quantify the number of TPs of v that quit the WAHN and r_V to quantify the number of TP service requests served by v during T_V: Therefore,

$$CCost^{TPEst-V} = (q_V + r_V) \times CCost^{TPEst} / 2$$

where (since the "TP Establishment" process is mainly based on the X.509 protocol):

$$CCost^{TPEst} \approx CCost^{X.509}$$

Let us use s_V to quantify the number of times that node v establishes keys with different nodes of the WAHN. Consider s'_V as the number of times *TP-V* acts as intermediary TP in the TKD protocol (during T_V). Theerfore, for the TKD protocol, the computational cost on node v can be calculated as:

$$CCost^{TKD-V} = (s_V + s'_V) \times CCost^{TKD} / (N_{AV} + 2)$$

and, for the X.509 protocol:

$$CCost^{X.509-V} = s \times CCost^{X.509} / 2,$$

Finally, the computational cost on *v* in HKMI and PKI solutions, respectively, is calculated as follows:

$$CCost^{HKMI-V} = CCost^{TPEst-V} + CCost^{TKD-V}$$

$$CCost^{PKI-V} = CCost^{X.509-V}$$

Consequently, from these two last equations, if $s_V + s'_V << N_{CEQ}$, HKMI is computationally "cheaper" on a node *v* than PKI for $q_V + r_V \leq s_V$. Here we have to distinguish to cases: (1) *v* is a NTP node; (2) *v* is a TP node.

In case (1), the above condition reduces to $q_V \leq s_V$. Assume that every t_q seconds a node quits the WAHN. Then, $q \approx T_V / t_q$ nodes leave the WAHN during T_V. Further assuming a uniform probability of node quitting, then $(N/P) \times q$ TP nodes quit during T_V. Similarly, the probability that a specific TP node (more concretely the one serving *v*) quits is 1/*N*. Then:

$$q_V = (T_V / t_q) \times 1/P$$

This equation shows that the period of time T_V that a node *v* can expect to be connected to the WAHN without the need to re-establish a new TP is $P \times t_q$. For instance, consider a WAHN with 256 nodes and with a (high) membership change rate of *t* = 10 seconds[16] (i.e. $t_q = 20$). In such a case, $T_V \approx 1.4$ hours.

Consequently, if during the whole period of time a NTP node *v* is connected to the WAHN, *v* establishes in average more than *one* new key using the TKD protocol each period T_V, HKMI results more computationally efficient for *v* than using PKI.

In case (2), the value of r_V can be controlled by *v*. In each interval of T_V seconds, *v* should not accept more TP service requests than the number of new keys to be established minus one, i.e. $s_V - 1 \geq r_V$. Consequently, if during the whole period of time a TP node *v* is connected to the WAHN, *v* establishes in average more than $r_V + 1$ new keys using the TKD protocol each time period T_V, HKMI results more computationally efficient for *v* than using PKI.

A last rule can be derived from the previous analysis: *nomadic* nodes, which can be foreseen to have a short transitive connection to the WAHN, should not play the role of TPs.

3.6.3.1 Assimilation of TP-domain

To minimize the computational overhead caused by "TP Establishment" process when a TP quits the network, TP-domains can be assimilated.

---

[16] The average membership change rate *t* is influenced by nodes joining and quitting the WAHN. If nodes join and quit at the same average rate $t_j = t_q$, then $t = t_q/2$.

Before a TP quits the WAHN, it transfers its trusted nodes to its parent TP. In this manner, nodes from the quitting TP become assimilated into the parent's TP domain without having to re-establish trust with any TP. For instance, if TP-Y on Figure 4 quits, nodes *v* and *z* formerly included in D_{TPY}, become assimilated by D_{TPX} (see Figure 11).

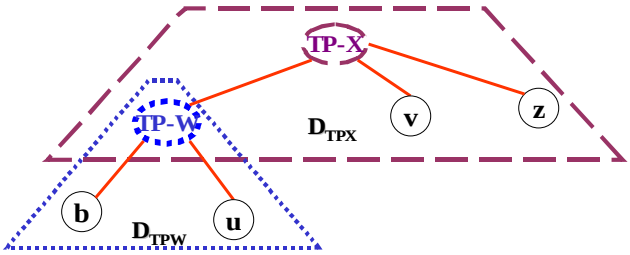


Figure 11. Assimilation of a TP-domain

Consequently, nodes only need to establish their own TP just in joining the network. Additionally, changes on WAHN membership do not increase the computational overhead of the HKMI significantly.

3.6.4 Storage Cost

Although we do not consider the storage requirements a relevant constrain in WAHNs of nodes with moderate resources, here we provide a brief analysis on the storage overhead of the HKMI.

The main storage requirements of the HKMI (aside from the need to store public keys and certificates) arise from the need to store a TP-Route table and TP-shared-keys on TP nodes.

Assuming a naïve implementation, the TP-Route table contains the IDs of the *P* nodes of the WAHN. The TP-shared-key can be computed using the method to compute long-term shared keys in Lotus Notes [16]. With this method the TP-shared-key $S_{Y,TPX}$ issued by *TP-X* to any node *y* included in D_{TPX} can be easily derived from a secret S_{TPX} (only known to *TP-X*) and the identifier ID_Y, without the need for *TP-X* to store $S_{Y,TPX}$, i.e. *TP-X* only needs to store its secret S_{TPX}.

Therefore, the storage cost on *TP-X* is calculated as:

$$SCost^X = P \times Size(ID) + Size(S_{TPX})$$

In WAHN applications using node IDs of 8 bytes and secrets of 16 bytes, each TP needs to merely store 2 Kbytes of data in order to enable up to *P* = 256 nodes in the WAHN.

3.6.5 Security Analysis

WAHN applications can be divided in two big cases depending on the ownership of the nodes:

1. Mobile nodes are owned by a single administrative entity (e.g. a hospital IT administration),

2. Mobile nodes are owned by different administrative entities (e.g. different human users).

From the security perspective we have to further distinguish two types of case (1.). In applications of type 1.A the risk of node compromise is low or null (Recall we designed the HKMI particularly for applications of type 1A). Type 1.B models applications with moderate or high risk of node compromise.

In applications of type 1.A, the major security risks are imposed by the open nature of wireless WAHNs. Here an outside attacker may try to anonymously eavesdrop, modify and replay WAHN wireless communications and/or even inject bogus messages.

In this case, the HKMI exhibits perfect security. In effect, the messages exchanged by core HKIM protocols are always transmitted through channels protected with confidentiality and integrity mechanisms.

In practice, employing an implementation with secure cryptographic algorithms and appropriate key lengths and lifetimes, no security sensitive information can be learnt eavesdropping wireless communications in offline or real-time attacks. Additionally, these messages cannot be replayed or modified and bogus messages cannot be injected in the wireless channel without going detected.

In applications of type 1.B an attacker may additionally compromise nodes. This attacker can fake trusted identities or issue false keys and trust recommendations.

Note that, in order to succeed in such an attack, first the attacker must capture a TP node. This task is particularly difficult because TPs remain anonymous to eavesdroppers. This feature "hides" the TPs to the attacker and, consequently, he/she needs to capture nodes randomly.

In a network of P nodes with N TPs the probability of success in capturing a TP node is N/P. Imagine $N/P = 0.1$. In this example, the attacker needs to capture an average of 10 nodes to succeed finding a TP among them. This lets the WAHN enough time to detect and react against the attack. In general, in applications of type 1.B, the ratio N/P should be high to minimize the risk of TP captures.

In addition to the threats of applications of type 1.B, in applications of case 2 some nodes may misbehave by not cooperating. Moreover, a misbehaving user may try to fake information in its behalf or in behalf of its “mates”. These kind of attacks are common in other security solutions based on node trust and cooperation [62][15][17], and, particularly, also in the HKMI.

In applications of case 2 the security robustness of the HKMI can be improved by adding one or a combination of the following countermeasures. The first countermeasure consists in coupling reputation mechanisms [65] with HKMI. As a misbehaving node is detected, it gets a "bad" reputation value. Nodes with bad reputation (too many associated "bad" reputation values) is not trusted anymore by the rest of nodes, despite it holds a public key certificate signed by a trusted CA.

Second, the average number of intermediate TP nodes can be minimized to reduce the risk that an attacker is among them. Similarly, the formation of isolated trust graph cycles (to the cost of decreased trust graph connectivity) can be allowed to reduce the effect of the attack to just a portion of the WAHN.

Finally, nodes can establish multiple alternative TPs (to the cost of increased computational overhead). In this manner, there exist multiple trust routes connecting two end nodes. The probability of having a misbehaving TP between two end nodes decreases when multiple alternative routes exist.

3.7 Summary and Conclusions

In this chapter, we have presented the HKMI for WAHN applications requiring efficient confidentiality, integrity, authentication and non-repudiation services. It finds particular application at WAHNs of moderate-resource nodes carried/wore or placed around human users.

We apply a cooperative approach to improve time and energy performance of key establishment protocols enabled by existing WAHN PKIs. The HKMI uses an underlying PKI to initially set-up trust relationships between nodes of the WAHN. In this manner, a random trust graph connecting all the nodes of the WAHN is generated. Nodes of the shortest trust path connecting two end nodes cooperate on demand to securely distribute trust information and symmetric keys to the end nodes.

Our analysis demonstrates that HKMI operating with a single CA allows nodes to establish keys up to two orders of magnitude faster (see Figure 8) and investing up to three orders of magnitude less battery resource (see Figure 6) than with PKI with a single CA.

Typically, in WAHNs nodes will not possess a certificate directly signed by a common CA but by a chain of multiple intermediate certificates leading to a common root CA. In such a scenario, the improvement of HKMI in respect to PKI increases further. To establish a symmetric key, in PKI two nodes need to exchange and validate multiple intermediate certificates to ultimately validate the certificate validating each other's public key. In HKMI intermediate TPs generate and distribute a symmetric key to both nodes without the need to exchange long public key-based messages or compute intensive PKC operations.

HKMI enables time and energy efficient security in WAHNs independently of the WAHN membership. However, because the “TP Establishment” process is based on public key cryptography, the same concept cannot be applied to ad hoc networks of extremely resource-constrained nodes (e.g. for wireless sensor networks).

Additionally, because trust and key distribution is based on cooperative trust, the HKMI does not result an optimal security solution for WAHNs of unattended nodes in public or hostile deployment areas. Finally, a cooperative key distribution approach may add significant energy and delay overhead in WAHNs of low-duty cycle and highly mobile nodes.

All these problems are tackled and solved in Chapter 4.

4 Deterministic Pairwise Key Pre-Distribution

Key pre-distribution is to date the best alternative to provide security for WAHNs of extremely resource-constrained nodes [30][2][3].

In this chapter, we present our DPKPS [2] for WAHN security. In short, the DPKPS is a system to pre-distribute keying material to WAHN nodes. It enables any pair of nodes to *directly* establish a pairwise symmetric key in an efficient manner.

This chapter is organized as follows. In Section 4.1 we set the scope, assumptions and objectives for the design of the DPKPS. Sections 4.2 and 4.3 introduce important theoretical background to understand the development of the DPKPS in Section 4.4. In subsection 4.4.5 we further describe how nodes shall be identified to substantially improve the communication efficiency of the original DPKPS. The exhaustive and thorough theoretical and practical analysis of the DPKPS is presented in Sections 4.5 and 4.6, respectively. Sections 4.7 and 4.8 compare the DPKPS with related work. Section 4.9 summarizes and concludes this chapter.

4.1 Design Scope, Assumptions and Objectives

The DPKPS is targeted for WAHN applications requiring extremely low-power security services of authentication, communication confidentiality and integrity. It is of especial interest for (but not limited to) mobile sensor network (MSN) applications.

Nodes are low-cost battery-powered devices with limited computational and storage capabilities. Representative devices include the AquisGrain wireless nodes (see Figure 16) developed by Philips [54][55] and the MICA Mote family of wireless motes developed by Crossbow.

An AquisGrain (and MICAz) is provided with an ATmega128L low-power 8-bit micro-controller and a 16-bit wide bus. It can be configured to work at a clock frequency of up to 8MHz. Because of these features intensive security operations in AquisGrain can take up to several seconds.

Aquisgrain (and MICAz) also includes a 128 Kbytes of flash memory, a 4 Kbytes EEPROM and a 4 Kbytes of SRAM. The EEPROM functions as a computer BIOS, which stores critical data to enable self-booting. The flash is mainly reserved to save application software. In sensor network applications, these nodes have intermittent connectivity to the WAHN. Consequently, a big portion of the flash memory must also be reserved to store transiently gathered sensor data. The SRAM offers just 4 Kbytes to store dynamic variables handled by running processes. Because of these memory limitations an AquisGrain cannot compute operations with big numbers as required for public key cryptography. Moreover, the footprint of security modules cannot be of an excessive size.

Aquisgrain is sourced to a tiny cell battery of 3 volt of limited energy capacity. In typical sensor network applications, it is required that nodes can operate weeks, months and even years without manual intervention. Therefore, usage of power-intensive operations is prohibitive.

To reduce manufacturing and WAHN-overall costs, these nodes are typically produced without tamper-protection modules.

In the applications considered in this chapter nodes are deployed and left independent in a given physical area. Because of different reasons (e.g. the human owner cannot or has difficult access to that area; the owner needs a fast deployment, and so forth) nodes are typically randomly deployed. After the deployment phase, i.e. when the nodes occupy a position in the final deployment area, nodes within wireless range –neighbors– form links autonomously and, ultimately, a WAHN. Because nodes operate at low-duty cycle, nodes rely on intermittent short-range wireless communication to sporadically connect to the WAHN [40].

After the initial deployment, new nodes may be added to the WAHN sometime later. Naturally, these nodes need to be interoperable with the old nodes existing at the WAHN.

In typical WAHN applications, nodes are attached to mobile objects (e.g. to a human body or to an animal). In this case, the process of node deployment and WAHN formation is dynamic. Thus, predicting which nodes will come into proximity and establish wireless links becomes even more difficult than in the previous model. In such applications, the best approach is assuming that any node may come into proximity and establish wireless links with any other node. Because of the singularity of this case, it is also possible that WAHNs coexist sparsed in the same deployment area. Mobile nodes randomly move and eventually connect to any of these WAHNs [2][3].

The WAHN can be formed of a huge number *N* of nodes. That is, up to hundreds of thousands of nodes. Because of intermittent wireless connectivity, multiple WAHN partitions can occur. In mobile scenarios, the *N* nodes disperse and form multiple non-interconnected not necessarily equally-sized WAHNs.

Typically nodes are deployed in public or hostile places and left unattended [29][40], where the risk of node captures becomes evident.

Our design aims at maximizing the scalability, connectivity and security properties of a key pre-distribution system while adapting to resource and energy limitations of nodes. The DPKPS must exhibit perfect connectivity property (i.e. probability *one* that two arbitrary nodes can establish a key) of the DPKPS independently of the WAHN membership, size, topology, physical connectivity and node density. Consequently, it can be equally applied to small/medium/large partitioned/unpartitioned WAHNs of static/mobile nodes.

4.2 λ-degree Symmetric Bivariate t-Polynomial Set

In the rest of the chapter, let us assume that the WAHN application requires keys of $\lfloor \log q \rfloor$ bits (e.g. 64 or 128 bits). We define $t = \lceil \lfloor \log q \rfloor / \lfloor \log q' \rfloor \rceil$, $t \geq 1$, and $N' = q' - 1$. Typically $N' << q - 1$.

A symmetric bivariate λ-degree polynomial over a finite field $F_{q'}$ (where q' is a prime) is a polynomial of the form $f(x, y) = \sum_{i,j=0}^{\lambda} a_{ij} x^i y^j$, where each of the coefficients a_{ij} are

randomly taken from a finite field with q' elements and $a_{ij} = a_{ji}$ (Note that $f(x, y) = f(y, x)$).

After Blundo et al [19], a polynomial share $f(p_u, y)$ is defined as the evaluation of $f(x, y)$ in a value p_u of $F_{q'}$. Polynomial evaluation involves modular addition and multiplication operations in $F_{q'}$. A partial pairwise key of $\lfloor \log q' \rfloor$ bits $K'_{uv} = f(p_v, p_u) = f(p_u, p_v)$ results from the evaluation of $f(p_u, y)$ at point p_v, or, symmetrically, from the evaluation of $f(p_v, y)$ at point p_u.

The resiliency α of the polynomial is $\alpha = \lambda + 1$, i.e. α distinct polynomial shares need to be pulled together to generate the original polynomial [19]. A polynomial can be used to generate up to $N' = q' - 1$ distinct polynomial share[17]s. Each polynomial share can be used to generate N' pairwise keys [19]. The length of a polynomial share $f(p_u, y)$ is $(\lambda + 1)\lfloor \log q' \rfloor$ bits.

Let us define a joint set of t λ-degree bivariate polynomials $\{f_i(x, y)\}_{i=1,\ldots t}$ with coefficients on $F_{q'}$ as a t-polynomial-set $F_i(x, y)$. The t-polynomial-set $F_i(p_u, y)$ evaluated at point p_u hereafter is a t-polynomial-set share.

After Liu and Ning[23], a pairwise key K_{uv} of target length $\lfloor \log q \rfloor$ bits (e.g. of 64 or 128 bits) can be compounded by concatenating the t partial keys $\{K_{uv,i}\}_{i=1\ldots t}$ generated with t polynomial shares $\{f_i(u, y)\}_{i=1,\ldots t}$ with coefficients on F_q' ($q' << q$, where q' is a prime of the form $q' = 2^k + 1$) without a significant loss of security. That is, the resulting $\lfloor \log q \rfloor$-bit key possesses similar entropy as it had been generated with a λ-degree bivariate polynomial with coefficients on F_q (with q prime).

Therefore, a t-polynomial-set $F_i(x, y)$ can be used to generate N' distinct t-polynomial-set shares and each t-polynomial-set share can be used to generate N' pairwise keys. A t-polynomial-set is λ-collusion resistant. The length of a t-polynomial-set share $F_i(u, y)$ is $(\lambda + 1)\lfloor \log q \rfloor$ bits.

The generation of keys using t-polynomial-sets can be applied to any polynomial-based KPS under certain lower bound on λ imposed by q, q' and the total number of polynomials over $F_{q'}$ used by the KPS. We will extend on this last issue in section 4.5.6. The value q' can be chosen to optimize computational efficiency in nodes (see section 4.6.5.5).

4.3 Finite Projective Planes

A *Balanced Incomplete Block Design* (BIBD) [48][49][50] is an arrangement of v distinct elements into w blocks such that each block contains exactly k distinct elements, each element occurs in exactly r different blocks, and every pair of distinct elements

[17] Although $F_{q'}$ contains q' elements, the value q' cannot be used for evaluating a polynomial because it represents the value $0 \bmod q'$, i.e. $f(q', y) = k$ is a constant that cannot be used to generate pairwise keys.

occurs together in exactly x blocks. The design can be expressed as (v,k,x), or equivalently (v,w,r,k,x), where: $x(v-1)=r(k-1)$ and $wk=vr$.

In a symmetric BIBD (SBIBD) $w=v$ and, thus, $k=r$. A SBIBD has four interesting properties: every block contains $k=r$ elements, every element occurs in $k=r$ blocks, every pair of elements occurs in x blocks and every pair of blocks intersects in x elements.

Finite Projective Planes (FPP) is a subset of SBIBDs of special interest for key pre-distribution. An FPP is an SPIBD with parameters $(n^2+n+1,n+1,1)$ [50][49]. An FPP exists for any prime power n, where n≥2. FPP of order n has four properties: (i) every block contains exactly n+1 elements, (ii) every element occurs on exactly n+1 blocks, (iii) there are exactly n^2+n+1 elements, and (iv) there are exactly n^2+n+1 blocks.

In the following, let B_i denote block number i of FPP-$(n^2+n+1,n+1,1)$. Let also $\{b_{i,1},b_{i,2},...b_{i,n+1}\}$ be the elements of B_i. Note that each element $b_{i,j}$, $j=1...n+1$, occurs in n additional blocks of the FPP.

Example 1: The seven blocks of an FPP-$(7,3,1)$ constructed with the integers from 1 to 7:

$$\{124\},\{235\},\{346\},\{457\},\{561\},\{672\},\{713\}$$

4.4 Deterministic Pairwise Key Pre-Distribution Concept

The DPKPS pre-distributes $n+1$ distinct t-polynomial-set shares $F_{b_{i,j}}(p_{u_j},y)$, $j=1...n+1$, to each node u. The indices $b_{i,j}$ of $F_{b_{i,j}}(p_{u_j},y)$, for $j=1...n+1$, are associated to the element $b_{i,j}$ of a block B_i of an FPP-$(n^2+n+1,n+1,1)$.

This distribution guarantees that any two nodes u and v carry distinct t-polynomial-set shares $F_k(p_u,y)$ and $F_k(p_v,y)$, respectively, of (at least) one common t-polynomial-set $F_k(x,y)$ and, thus, can establish a pairwise key K_{uv}.

The DPKPS consists of four stages:

1. *Set-Up.* A number of t-polynomial-sets are generated and evaluated for accommodating up to N nodes,
2. *t-Polynomial-Set Shares Pre-distribution.* A number of t-polynomial-set shares are pre-distributed to each node,
3. *t-Polynomial-Set Shares Discovery*. Two nodes find that they carry shares of the same t-polynomial-set,
4. *Pairwise Key Establishment.* Two nodes establish a pairwise key.

4.4.1 Set-up

A set-up server randomly generates a set $\mathfrak{I}$ of $t\times(n^2+n+1)$ λ-degree bivariate polynomials $\{f_b^r(x,y)\}_{b=1...n^2+n+1}^{r=1...t}$ over F_q. Subsequently, for $b=1...n^2+n+1$, the set-up

server sequentially picks *t* polynomials from $\mathfrak{I}$, and forms n^2+n+1 *t*-polynomials-sets $F_b(x,y)$.

Then, the set-up server generates an FPP-$(n^2+n+1, n+1, 1)$, with elements belonging the set *S* of integers from 1 to n^2+n+1. The set *S* is associated with a *t-polynomial-set pool*, i.e. each element *b*, $b=1...n^2+n+1$, in *S* is associated with a distinct t-polynomial-set $F_b(x,y)$.

Further, each block B_i of FPP is associated with $N'/(n+1)$ *rings* of distinct t-polynomial-set shares. The properties of FPPs guarantee that any pair of rings has at least one t-polynomial-set $F_k(x,y)$ in common.

Example 2: Seven t-polynomial-sets can be arranged in blocks of an FPP-$(7,3,1)$ as follows:

$$\{F_1F_2F_4\},\{F_2F_3F_5\},\{F_3F_4F_6\},\{F_4F_5F_7\},\{F_1F_5F_6\},\{F_2F_6F_7\},\{F_1F_3F_7\}$$

4.4.2 t-Polynomial-Set Shares Pre-Distribution

Each node *u* of an *N*-node population receives from the set-up server a distinct ring, which includes $n+1$ *t*-polynomial-set shares $F_{b_{i,j}}(p_{u,j},y)$, where $p_{u,j} \in F_{q'}$, $b_{i,j} \in B_i \in$ FPP, and $j=1...n+1$.

To guarantee uniqueness of pairwise keys, two different nodes *u* and *v* cannot receive the *same* *t*-polynomial-set $F_k(x,y)$ evaluated in the same point p_k. Since each *t*-polynomial-set $F_b(x,y)$, $b=1...n^2+n+1$, can be evaluated in $N'=q'-1$ different points and the index *b* of $F_b(x,y)$ appears in $n+1$ FPP blocks, then each of these blocks (where *b* occurs) shall be used to pre-distribute *distinct* shares of $F_b(x,y)$ to no more than $N'/(n+1)$ different nodes.

Here we detail a key pre-distribution process to accommodate up to $N'(n^2+n+1)/(n+1)$ nodes[18]:

1. Starting with the first block B_1 of FPP with elements $\{b_{1,1},...b_{1,n+1}\}$, the first node (u_1) receives t-polynomials-set shares $F_{b_{1,1}}(p_1,y)$ to $F_{b_{1,n+1}}(p_1,y)$ evaluated at point p_1 of $F_{q'}$; the second node (u_2) receives $F_{b_{1,1}}(p_2,y)$ to $F_{b_{1,n+1}}(p_2,y)$ evaluated at point p_2; and, so forth; till the $N'/(n+1)$-th node ($u_{N'/(n+1)}$) receives $F_{b_{1,1}}(p_{N'/(n+1)},y)$ to $F_{b_{1,n+1}}(p_{N'/(n+1)},y)$ evaluated at point $p_{N'/(n+1)}$,
2. Following with the second block B_2 of FPP with elements $\{b_{2,1},...b_{2,n+1}\}$, assume $b_{1,1}=b_{2,1}$, node $u_{1+N'/(n+1)}$ receives $F_{b_{1,1}}(p_{1+N'/(n+1)},y)$, which is evaluated at point

[18] A variation of this process can be used to accommodate $N \le N'(n^2+n+1)/(n+1)$ nodes, if desired.

$p_{1+N'/(n+1)}$ (because $F_{b_{1,1}}(x, y)$ is already evaluated in lower points for nodes u_1… $u_{N'/(n+1)}$), and $F_{b_{2,2}}(p_1, y)$ to $F_{b_{2,n+1}}(p_1, y)$ evaluated at point p_1; and so forth.

This process is repeated to accommodate up to N nodes using all the blocks of the FPP.

Example 3. Let us use the finite field F_{257} with elements 1,2,...257 ($N' = 256$) and the FPP-$(7,3,1)$ of examples above. In this case, the t-polynomial-set share pre-distribution phase goes as follows (see Figure 12): starting with $B_1 = \{124\}$, the set-up server distributes to node 1 (u_1) $F_1(1, y)$, $F_2(1, y)$ and $F_4(1, y)$; node 2 (u_2), receives $F_1(2, y)$, $F_2(2, y)$ and $F_4(2, y)$; and, so forth, till node 85 ($u_{N'/(n+1)}$) receives $F_1(85, y)$, $F_2(85, y)$ and $F_4(85, y)$.

Following with $B_2 = \{235\}$, node 86 receives $F_2(86, y)$, $F_3(1, y)$ and $F_5(1, y)$; node 87 receives $F_2(87, y)$, $F_3(2, y)$ and $F_5(2, y)$ and so forth.

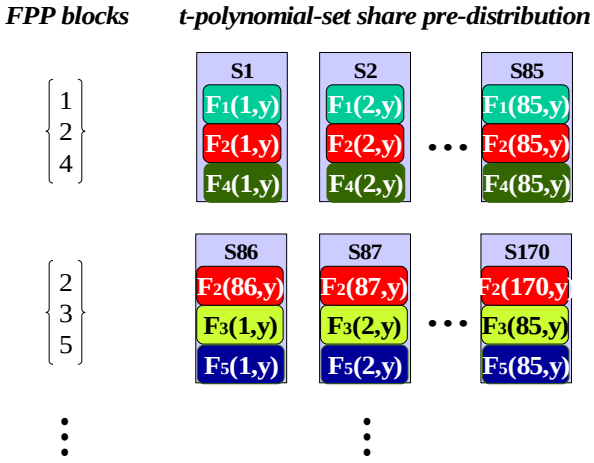


Figure 12. Example of t-polynomial-set pre-distribution

A simple way to identify a node u in the DPKPS is to concatenate the n+1 indices $b_{i,1},...b_{i,n+1}$ of the n+1 t-polynomial-set shares it carries with the n+1 points $p_{u1}...p_{un+1}$ where they are evaluated. Such an ID uniquely identifies a node u and enables simple discovery of common t-polynomial-set shares.

Because a simple ID requires excessive storage space and incurs excessive communication overhead (see Sections 4.5.3 and 4.5.4), we discuss an improved version of node IDs in section 4.4.5.

4.4.3 t-Polynomial-Set Shares Discovery

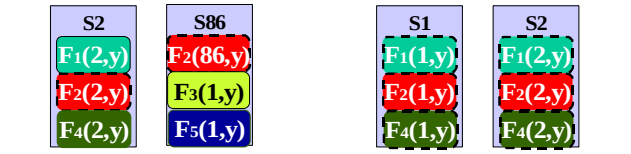
After node deployment, before establishing a pairwise key, each node u must discover which t-polynomial-set it shares with its communication neighbor v.

Nodes u and v exchange their IDs, which implicitly contain the indices of the n+1 t-polynomial-set shares they carry and the points $p_{u,1}...p_{u,n+1}$, $p_{v,1}...p_{v,n+1}$ where their respective n+1 t-polynomial-set shares are evaluated.

Finally, they find an index k (corresponding to the common t-polynomial-set $F_k(x,y)$) and the respective evaluation points p_u and p_v.

Because of the properties of FPPs, two nodes u and v with respective $n+1$ t-polynomial-set shares distributed according to the elements of different blocks $B_i, B_j \in$ FPP, $i \neq j$, carry shares of *one* t-polynomial-set $F_k(x,y)$ (See Case A on Figure 13).

Similarly, two nodes with respective $n+1$ t-polynomial-set shares distributed according to the elements of the same block $B_i \in$ FPP carry shares of $n+1$ common t-polynomial-sets (See Case B on Figure 13). This second kind of nodes can use any (or a combination) of the $n+1$ t-polynomial-set shares to compute a unique pairwise key of $\lfloor \log q \rfloor$ bits.



Case A: One common share Case B: n+1 common shares

Figure 13. t-polynomial-set share discovery cases

4.4.4 Pairwise Key Establishment

To calculate a pairwise key K_{uv}, node u firstly evaluates the t λ-degree bivariate polynomials $f_k^r(p_u, y)$ (included in $F_k(p_u, y)$), for $r = 1...t$, at point p_v (i.e. $f_j^r(p_u, p_v)$) to obtain t partial keys.

Finally, node u truncates the t partial keys to $\lfloor \log q' \rfloor$ bits and concatenates the t key segments to form a final pairwise key K_{uv} of $\lfloor \log q \rfloor$ bits.

Example 4: Following with Example 3., to establish a pairwise key $k_{1,87}$, nodes 1 and 87 evaluate $F_2(1, y)$ and $F_2(87, y)$ at points 87 and 1, respectively.

4.4.5 Optimized Node Identifier

Because using simple node IDs considerably augments the storage and communication costs of the DPKPS (see Sections 4.5.3 and 4.5.4), in this section we present an alternative method.

The method to construct optimized node ID exploits properties of FPPs based on *mutually orthogonal latin squares* (MOLS) [48][49][53].

The DPKPS FPP-$\left(n^2+n+1, n+1, 1\right)$ is to be developed from a set of n-1 MOLS of order n.

A node u is to be identified concatenating three numbers i, i_p, and l^{e_m}, where $1 \le i \le n^2+n+1$, $1 \le i_p \le N'/(n+1)$ and $1 \le e_m \le n$. The first number, i, identifies a block $B_i \in$ FPP according to which the t-polynomial-set shares of u are chosen. The

second number, i_p, identifies the turn i_p in the distribution of t-polynomial-set shares to u within B_i. The third number, l^{e_m}, identifies an element of the Latin square L_i from which B_i is derived.

The rest of stages are to be implemented as previously explained in sections 4.4.1, 4.4.2, 4.4.3 and 4.4.4.

The following subsections show that using such an FPP enables the optimized node IDs for the DPKPS without any loss of functionality. That is, the new ID also enables the discovery of t-polynomial-set shares and its points of evaluation.

In the performance analysis (Sections 4.5 and 4.6), we demonstrate the improved performance efficiency of the DPKPS processes enabled by the optimized node ID.

4.4.5.1 MOLS

A Latin square[19] is an $n \times n$ square matrix L whose entries consist of n elements such that each element appears exactly once in each row and each column [53].

We will use as elements of L the integers from 1 to n. A simple way to construct L is by placing the integers 1,2,...n in their natural order in the first row and, for consecutive rows, by cyclically rotating the previous row to the left. For instance, for $n = 3$:

$$L = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}$$

Two Latin Squares $L^1 = \left| l_{ij}^1 \right|$ and $L^2 = \left| l_{ij}^2 \right|$ on n elements 1,2,...n are *orthogonal* if, when superimposed, each of the n^2 ordered pair of elements $\left(l_{ij}^1, l_{ij}^2\right)$, $i = 1,2...n$; $j = 1,2...n$, occurs exactly once.

A set of Latin squares L^1, L^2 ,… L^t of the same order n, each of which is an orthogonal mate of each of the others, is called a set of MOLS. A set of n-1 MOLS of order n is a complete set [53].

$$L^1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}; L^2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \\ 2 & 3 & 1 \end{pmatrix}$$

For n a prime power the set of polynomials of the form $f_a(x, y) = ax + y$, $a \neq 0 \in F_n$ represent a complete set of n-1 MOLS of order n [53]. This leads to a simple construction method: let e_1, e_2, ...e_n be the elements of F_n, i.e. the integers 1…n. Then, for each element e_m, $m = 1,2,...n$, the elements $l_{ij}^{e_m}$ of the matrix $L^{e_m} = \left| l_{ij}^{e_m} \right|$ are *sequentially* calculated by the following equation:

---

[19] Also used to derive the popular Sudoku puzzles.

$$l_{ij}^{e_m} = (e_m \times e_i) + e_j \qquad \text{(Eq-1)}$$

Observe that the parameters n and e_m are sufficient to reconstruct a specific orthogonal Latin square $L^{e_m} = \left| l_{ij}^{e_m} \right|$. This is a property that we apply for the discovery of common t-polynomial-set shares (see Section 4.4.5.3).

4.4.5.2 Construction of an FPP from MOLS

Let L^1, L^2, … L^{n-1} be a complete set of MOLS of order n and M an $n \times n$ matrix. Firstly, matrix M is to be constructed by placing the n^2 integers 1… n^2 in their natural order from the first to the n-th rows. For instance, for n = 3:

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Secondly, generate an Affine Plane $\mathrm{AG}(2,n)$ of order n from the MOLS as follows [53]: (i) the first n blocks are the rows of M, (ii) the second n blocks are the columns of M, and (iii) the remaining n^2-n blocks are formed by sequentially superimposing each L^{e_m} on M, and taking as blocks the elements of M which correspond to a single element l^{e_m} in each L^{e_m}. Since each L^{e_m} contains n different elements, each $L^{e_m} : M$ superposition yields n blocks.

Finally, to obtain an FPP-$(n^2+n+1, n+1, 1)$, (i) add a new integer n^2+1 to the first n blocks of the Affine Plane, (ii) add a new integer n^2+2 to the second n blocks, (iii) add an integer n^2+2+e_m to the n blocks constructed from each L^{e_m}, and (iv) add a new block to the design, which contains the n+1 new added integers.

Note that, given n and an FPP block index i, it is simple to reconstruct the elements of block $B_i \in$ FPP, $1 < i \le 2n$. For instance, for n=3, block B_4 is constructed from the first column of M 3×3 and the integer 11, i.e. $B_4 = (1,4,7,11)$.

For a block $B_i \in$ FPP, $2n < i \le n^2 + n$, index i also implicitly identifies the index e_m, $1 \le e_m \le n-1$, of the Latin square L^{e_m}, from which B_i is generated. For instance, for n=3, block B_{12} is generated from L^2.

To reconstruct one of these blocks B_i, $2n < i \le n^2 + n$, the element l^{e_m} is additionally required.

4.4.5.3 Discovery of Common t-Polynomial-Set Share

In this section, we describe how a node u can calculate the indices of its own t-polynomial-set shares and those of a partner node v. By comparing this information, node u can derive the index k, $1 \le k \le n^2 + n + 1$, of the common t-polynomial-set share $F_k(p_u, y)$ with node v.

Recall that the indices of the t-polynomial-set shares $F_{b_{i,j}}(p_{u_j}, y)$, $j = 1...n+1$, which a node *u* carries, are a one-to-one mapping to the elements $\{b_{i,1}, b_{i,2}, ...b_{i,n+1}\}$ of a $B_i \in$ FPP.

As we advanced in the previous section given *n*, the index *i* of B_i and an integer l^{e_m}, it is possible to uniquely reconstruct $\{b_{i,1}, b_{i,2}, ...b_{i,n+1}\} = B_i$. Here we have to distinguish two cases:

1. blocks B_i, $1 \le i \le 2n$, and B_{n^2+n+1}, whose reconstruction is trivial;
2. blocks B_i, $2n < i \le n^2 + n$, whose reconstruction is also simple but requires the further analysis.

In the second case, we know by step (iii) of the construction of Affine Planes that the elements $\{b_{i,1}, b_{i,2}, ...b_{i,n+1}\}$ of B_i are taken from the positions at *M* marked by 2*n* coordinates $\{(i_1, j_1), (i_2, j_2), ...(i_n, j_n)\}$ where l^{e_m} occurs within L^{e_m}. Thus, determining these coordinates, a node gets the elements of B_i.

First, from the index *i* it is straightforward to derive e_m, which identifies the Latin square L^{e_m} used to pick *n* of the elements of B_i. Second, recall that in a Latin square each element appears exactly once in each row and each column. Then, l^{e_m} occurs only once in each row of L^{e_m}, i.e. l^{e_m} occurs at the positions $\{(1, j_1), (2, j_2), ...(n, j_n)\}$. Since l^{e_m} and e_m are known, using equation Eq-1, we have:

$$l^{e_m} = (e_m \times 1) + e_{j_1}, e_{i_1} = 1$$

$$l^{e_m} = (e_m \times 2) + e_{j_2}, e_{i_2} = 2$$

$$\dots$$

$$l^{e_m} = (e_m \times n) + e_{j_n}, e_{i_n} = n$$

Observe that $e_{j_n} = l^{e_m}$ because $(e_m \times n) = 0$ in F_n. This allows us to re-write the previous system of equations as follows:

$$e_{j_{k-1}} = e_{j_k} + e_m, \; k = n, n-1, ..., 2 \text{ and } e_{j_n} = l^{e_m} \in F_n \text{ (see note}^{20}\text{)}$$

Because $e_{j_{k-1}} = e_{j_k} + e_m \le 2n$, where $e_{j_k} \in F_n$, the value $e_{j_{k-1}}$ in F_n is calculated without the need of any division operation as follows:

---

[20] $l^{e_m} \in F_n$ because $l^{e_m} \le n$.

$$e_{jk-1} = \begin{cases} e_{jk-1} \Leftrightarrow e_{jk-1} \le n \\ e_{jk-1} - n \Leftrightarrow n < e_{jk-1} \le 2n \end{cases}$$

Now sequentially order the n distinct values $e_{jk} \in F_n$, $k = 1...n$, in a vector V of n positions $(e_{j_1}, e_{j_2}, e_{j_n})$.

Recall (from section 4.4.5.1) that the elements e_1, e_2, ...e_n from F_n are sequentially used to calculate each element $l_{ij}^{e_m}$ of the matrix $L^{e_m} = \left| l_{ij}^{e_m} \right|$, i.e. element e_1 is used to calculate the elements at the positions $\{(1,1),(2,1),...(n,1)\}$, element e_2 is used to calculate the elements at the positions $\{(1,2),(2,2),(3,2),...(n,2)\}$ and so forth. Consequently, each value $(e_{j_1}, e_{j_2}, e_{j_n}) \in F_n$ determines the coordinates $\{(1, j_1),(2, j_2),...(n, j_n)\}$ where l^{e_m} occurs within L^{e_m}, e.g. if for $i_3 = 3$, $e_{j_3} = 2$, then $j_3 = 2$ (l^{e_m} occurs at $\{(1, j_1),(2, j_2),(3,2)...(n, j_n)\}$).

Finally, mapping these coordinates to elements of matrix M, it is now straightforward to determine n out of $n+1$ the elements of B_i. Now, we have a block of the Affine Plane. Adding the integer $n^2 + 2 + e_m$ to the block, we get the block $\{b_{i,1}, b_{i,2},...b_{i,n+1}\} = B_i$.

4.4.5.4 Derivation of t-Polynomial-Set Share Evaluation Point

Assume a share of t-polynomial-set $F_k(x, y)$ has been distributed to v, i.e. $F_k(p_v, y)$. To derive the point p_v, where node u must evaluate its share $F_k(p_u, y)$ to generate a key K_{uv}, node u must follow a simple process enabled by the properties of the FPP.

From v's ID, node u knows the index k of the common t-polynomial-set, a block index i and the distribution order i_p, and the order n of the FPP. Recall i_p is the order of a node v in the distribution of t-polynomial-set shares according to block $B_i \in$ FPP, $1 \le i \le n^2 + n + 1$.

Let us advance that $p_v = s_k \lfloor N'/(n+1) \rfloor + i_p$, where s_k quantifies the number of occurrences of $F_k(x, y)$ in blocks $B_j \in$ FPP, $j = 1...i \le n^2 + n + 1$.

Because FPP is constructed from MOLS, its first n^2 elements occur *once* in each group of n subsequent blocks $B_{1+t}, B_{2+t},...B_{n+t}$, $t = 0, n, 2n, 3n...n^2$. Then, given a block index i, $1 \le i \le n^2 + n$, and a t-polynomial-set index k, $k \le n^2$, deriving its occurrence counter s_k is trivial, i.e. $s_k = \lfloor i/n \rfloor$.

Furthermore, each element of the form $k = n^2 + j$, $j = 1...n+1$, occurs n times in the group of blocks $B_{i+n(j-1)}, i = 1...n$. In this case, $s_k = i - n(j-1)$.

Finally, the elements $k = n^2 + j$, $j = 1...n+1$, of block B_{n^2+n+1} occur for the n+1-th time within the FPP.

4.5 Analytical Performance Evaluation

In this section we *analytically*[21] assess the DPKPS and compare it with related schemes.

4.5.1 Scalability

Each block of FPP is used to accommodate up to $N'/(n+1)$ different nodes and there are n^2+n+1 distinct blocks. Therefore, the number of nodes *N* that the DPKPS scheme can accommodate is given by:

$$N \le N'(n^2+n+1)/(n+1) \qquad \text{(Eq-2)}$$

4.5.2 DPKPS Resiliency

The resiliency α of the DPKPS can be calculated as the number of nodes that a smart attacker needs to capture to compromise all the n^2+n+1 t-polynomial-sets, i.e. to disrupt DPKPS security completely.

Consider a first case where each t-polynomial-set of the DPKPS is used for no more than λ nodes. In this case, the DPKPS exhibits perfect-resiliency (but the scalability is restricted to $N \le \lambda(n^2+n+1)/(n+1)$ (see Section 4.5.1).

Now consider that each t-polynomial-set is used to pre-distribute *i* shares, $\lambda+1 \le i \le N'$. The attacker succeeds compromising the n^2+n+1 t-polynomial-sets by getting $\lambda+1$ shares $F_b(p_{b,1},y)...F_b(p_{b,\lambda+1},y)$ evaluated at different points $p_{b,j} \in Fq'$, $p_{b,j} \neq p_{b,k}$, $j \neq k$, for $j=1...\lambda+1$, for each t-polynomial-set $F_b(x,y)$, $b=1...n^2+n+1$.

Two important properties of FPPs are used to further analyse the resiliency of the DPKPS:

1. Any $n+1$ blocks with a *specific element s* in common contain the n^2+n+1 elements of the set *S*.

   Demonstration. Any specific element *s* appears in exactly $n+1$ blocks of the FPP and any two blocks of the FPP shares just one element in common [50]. Then, in $n+1$ blocks with element *s* in common, the number of no common elements is $n(n+1)$. By adding element *s*, we obtain the n^2+n+1 elements of the set *S*.

2. And, as a consequence of property 1., there are n^2 blocks not containing a *specific element s* in common.

Let $\lambda+1 \le i/(n+1)$. Because t-polynomial-set shares are pre-distributed according to an FPP, the *smart attacker*[22] needs to capture $(\lambda+1)(n+1)$ nodes with the same *specific t-polynomial-set* $F_s(x,y)$ to be able to recover all the keys.

---

[21] A practical evaluation is also presented in a later section.

To compromise $F_s(x,y)$, a smart attacker needs to find λ+1 nodes carrying it. By choosing $F_s(x,y)$ and capturing n+1 nodes carrying shares of it, in rings associated to n+1 *different* FPP blocks, the wise attacker can compromise $F_s(x,y)$. In turn, he/she additionally gets n^2+n t-polynomial-set shares $F_b(p_b,y)$, $b=1...n^2+n+1$, $s \neq b$, evaluated at points $p_b \in Fq'$.

By repeating the previous operation λ+1 times, capturing n+1 different nodes each time, the smart attacker gets $(\lambda+1)\times(n^2+n+1)$ t-polynomial-set shares $F_b(p_{b,j},y)$, $p_{b,i} \in Fq'$, $p_{b,j} \neq p_{b,k}$, $j \neq k$, $j=1...\lambda+1$, $b=1...n^2+n+1$, and, thus, he can re-generate the n^2+n+1 t-polynomial-sets.

In case $\lambda+1 > i/(n+1)$, the smart attacker needs to capture a slightly increased number of nodes. In any case, the resiliency parameter (for the smart attacker case) can be approximated as[23]:

$$\alpha^{smart} = (\lambda+1)(n+1) \qquad \text{(Eq-3)}$$

Let us hereafter analyze the resiliency of the DPKPS under an *unlucky oblivious*[24] *attacker*.

Assume each t-polynomial-set is used to pre-distribute i shares, $\lambda+1 \leq i \leq N'$ According to the n+1 blocks of the FPP with the same *specific element s*, i shares of $F_s(x,y)$ and $i/(n+1)$ shares of each $F_b(p_b,y)$, $b=1...n^2+n+1$, $s \neq b$, are pre-distributed.

Then, according to the n^2 blocks of the FPP without the same *specific element s* $i-i/(n+1)$shares of each $F_b(p_b,y)$, $b=1...n^2+n+1$, $s \neq b$, are pre-distributed. Note that the total number of shares distributed using the n^2 blocks of the FPP without the same *specific element s* is then $(n^2+n)(i-i/(n+1))$.

Since every node carries n+1 distinct shares, the number of nodes carrying these shares is $(n^2+n)(i-i/(n+1))/(n+1)$.

Then, an unlucky oblivious attacker may need to capture up to $(n^2+n)(i-i/(n+1))/(n+1)+\lambda+1$ nodes to compromise the DPKPS. Therefore, in the case of the *unlucky* oblivious attacker, the resiliency parameter is:

$$\alpha^{unlucky} = i\,n^2/(n+1)+\lambda+1$$

In general, the resiliency parameter for a generic oblivious attacker is:

---

[22] Recall that the smart attacker captures nodes selectively (see Chapter 2).

[23] We will use $\alpha=(\lambda+1)(n+1)$ in the rest of the chapter.

[24] Recall that the oblivious attacker captures nodes randomly (see Chapter 2).

$$\alpha^{smart} \leq \alpha^{oblivious} \leq \alpha^{unlucky}$$

More precisely, the resiliency of the DPKPS can be calculated as the fraction of total communications compromised when an oblivious attacker captures N_c nodes randomly, where $N_c > \lambda$.

Consider two arbitrary non-captured nodes *u* and *v*, which share one t-polynomial-set $F_s(x,y)$. Since at least $\lambda + 1$ nodes carrying $F_s(x,y)$ need to be captured to compromise $F_s(x,y)$, the probability p_c that the oblivious attacker can compromise the link between *u* and *v* is:

$$p_c = 1 - \frac{\sum_{k=0}^{\lambda} \binom{i}{k} \binom{N-i}{N_c-k}}{\binom{N}{N_c}}$$

The equation of p_c can be used to calculate the fraction of total communications compromised when N_c nodes are captured.

To analyze how the parameters λ and *n* influence the resiliency of the DPKPS, we calculated the fraction of compromised communications as the number of compromised nodes grows (see Figure 14 where we considered a population of $N = 1000$ nodes and available memory space *m* equivalent to 125 keys).

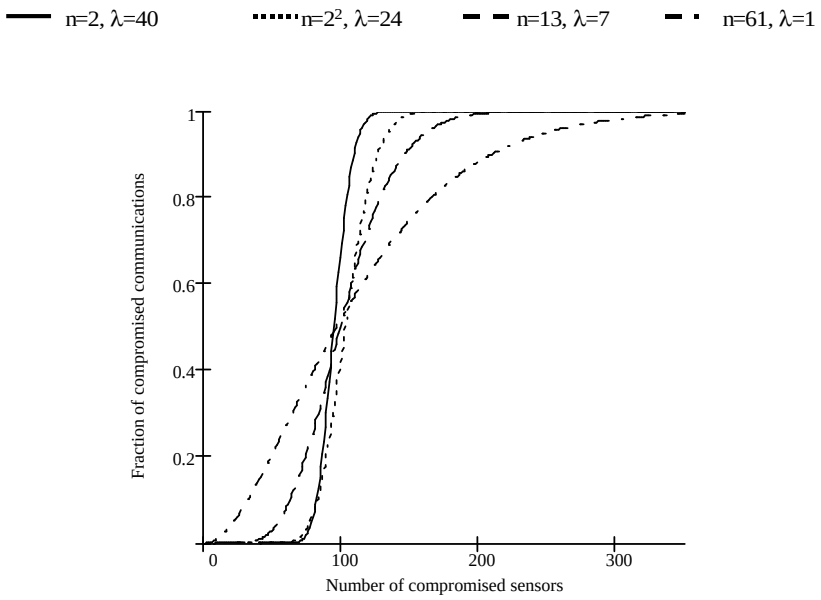


Figure 14: Fraction of compromised communications under oblivious attacker node captures ($N = 1000$, $m = 125$)

Figure 14 shows that the DPKPS exhibits a relatively long period of perfect resiliency for low values of *n* (and for relatively high values of λ). In these cases however, the transition from perfect resiliency to no resiliency rushes significantly around values of N_c close to *m*.

Consequently, configuration of the DPKPS with low value of n should be used in WAHN applications with high risk of node captures and relatively low scalability demands.

Figure 14 also shows that a DPKPS with relatively high value of n exhibits a smooth transition from perfect resiliency to no resiliency when N_c grows. In these cases, the attacker needs to capture much more than m nodes to compromise 100% of the DPKPS t-polynomial-sets (Recall, for comparison, that the resiliency of Blundo KPS is limited to m nodes). However, in these cases, the period of perfect resiliency is almost inexistent.

Consequently, this kind of configuration should be used in applications with low risk of node captures demanding huge scalability and computational efficiency. Additionally, it can be used in applications where nodes are provided of reliable tamper-resistant modules or where a certain fraction of compromised communications is tolerated.

In the rest of the chapter, unless otherwise stated, α denotes the resiliency of a KPS against the smart attacker.

4.5.3 Storage Cost

Each node needs to store n+1 t-polynomial-set shares, which include t λ-degree bivariate polynomial shares over $F_{q'}$.

Each polynomial share over $F_{q'}$ occupies λ+1 times the size of a partial key ($\log q'$ bits) and t of these polynomials λ+1 times the size of a pairwise key ($\log q$ bits).

Then,

$$m = (\lambda + 1)(n + 1) \qquad \text{(Eq-4)}$$

Additionally, each node must store its identifier. Let us analyze both the simple and the optimized node ID methods.

In simple ID, a node ID is constructed by concatenating the identifiers of the n+1 t-polynomial-set shares a node carries with the n+1 points in $F_{q'}$ where they are evaluated. The index of a t-polynomials-set can be codified in $\lceil \log(n^2 + n + 1) \rceil$ bits and a point of evaluation in $\log(q'-1)$ bits.

Therefore, the size of a simple node ID in bits is given by:

$$(n+1)\left[\lceil \log(q'-1) \rceil + \lceil \log(n^2 + n + 1) \rceil\right] \qquad \text{(Eq-5)}$$

The storage overhead contributed by the simple node ID is of significant relevance as n grows (see Figure 15), especially when compared to the memory size m in nodes. For instance, with $\log q' = 8$ and $n = 61$, then (by applying equation Eq-5) $size(ID) = 1240$ bits. This ID size is equivalent to a 15.5 % of the memory occupied by t-polynomial-set shares when a memory $m = 125$ (8000 bits) is considered.

The optimized node ID is constructed by concatenating the identifier *i* of a block $B_i \in$ FPP, with a number l^{e_m}, identifying an element within a Latin square L^{e_m}, and with the number i_p identifying the turn within B_i in the distribution of t-polynomial-set shares to *u*.

Observe that the maximum number of different blocks $B_i \in$ FPP is n^2+n+1 and the maximum number of different nodes that can be accommodated following each block is $N'/(n+1)$. Additionally, the maximum number of different elements within a Latin Square L^{e_m} of order *n* is *n*.

Thus, the size of the optimized node ID is:

$$\lceil \log(n^2+n+1) \rceil + \lceil \log((q'-1)/(n+1)) \rceil + \lceil \log(n) \rceil$$

For the example previously mentioned, $size(optID) = 19$ bits, which just corresponds to a 0.95% of the memory $m = 125$.

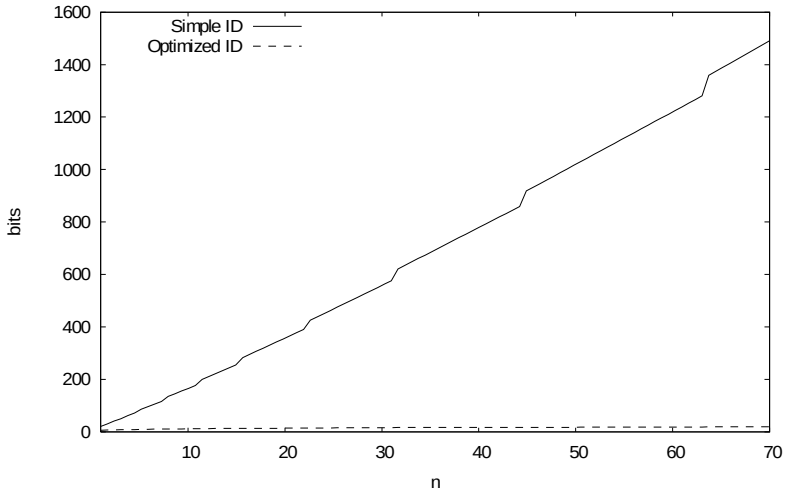


Figure 15 Simple and optimized node IDs length

4.5.4 Communication Cost

The DPKPS needs to find which common t-polynomial-set share nodes *u* and *v* have in common. Note that, compared to other related KPSs [20]-[24], the DPKPS does not involve third-party nodes in the establishment of a common key K_{uv}.

The rest of stages of the DPKPS do not demand further exchange of information between nodes.

To establish a pairwise key in the DPKPS two arbitrary nodes *u* and *v* need to exchange the *n*+1 indices of their t-polynomial-set shares and the point where the common share is evaluated. This is simply achieved by exchanging nodes IDs.

Using the simple method to generate IDs, both nodes need to communicate $2 \times (n+1) \times [\lceil \log(n^2+n+1) \rceil + \lceil \log(q'-1) \rceil]$ bits.

And, using the optimized node ID method, both nodes need to communicate $\left(\lceil \log\left(n^2+n+1\right)\rceil + \lceil \log\left((q'-1)/(n+1)\right)\rceil + \lceil \log(n)\rceil\right)\times 2$ bits.

4.5.5 Computational Cost

If simple node IDs are used in the DPKPS, only the generation of pairwise keys needs of computational effort from node nodes.

To establish a pairwise key, a node needs to evaluate a *t*-polynomial-set share at the corresponding point of evaluation. The evaluation of a polynomial share over $F_{q'}$ requires λ modular multiplications and λ modular additions in $F_{q'}$ [19]. A *t*-polynomial-set includes *t* polynomial shares over $F_{q'}$.

Therefore, the computational cost of generating a pairwise key can be approximated as:

$$CCost^{KeyGen} = t \times \lambda \times \left(add\_F_{q'} + mult\_F_{q'}\right)$$

If optimized node IDs are used, then nodes need to additionally compute operations to discover a common t-polynomial-set share and to derive the point of evaluation. First, generating a block of FPP from MOLS requires (at most) *n* additions and *n* multiplications in F_n. To find a common t-polynomial-set share, a node needs to generate and compare two FPP blocks.

Then, neglecting the comparison operations, the computational cost of finding a common t-polynomial-set is approximated as:

$$CCost^{ShareDiscovery} = 2n \times \left(add\_F_n + mult\_F_n\right)$$

To find the t-polynomial-set share point of evaluation, a node needs to evaluate the formula $p_v = s_k \lfloor N'/(n+1) \rfloor + i_p$ with normal arithmetic, where, depending a FPP block index *i* and a t-polynomial-set index *k*, $s_k = \lfloor i/n \rfloor$, $s_k = i - n(j-1)$, or $s_k = n+1$ (refer to Section 4.4.5.4).

Then, for each case, the computational cost of finding a point of evaluation is approximated as:

$$CCost^{EvalPoint\ Derivation} = 2add + 2div + mult \quad (\text{case } s_k = \lfloor i/n \rfloor)$$

$$CCost^{EvalPoint\ Derivation} = 4add + div + 2mult \quad (\text{case } s_k = i - n(j-1))$$

$$CCost^{EvalPoint\ Derivation} = 3add + div + mult \quad (\text{case } s_k = n+1)$$

We demonstrate later in our practical analysis that $CCost^{ShareDiscovery}$ and $CCost^{EvalPoint\ Derivation}$ are insignificant in practice (see Section 4.6.5).

4.5.6 Existence and Security of Polynomials

We cannot make *q'* arbitrarily small (to adapt to the computational restrictions of low-bit CPUs) and n^2+n+1 arbitrarily large (to accommodate more nodes) without paying

off a decrease in security strength of the DPKPS (in general, of any KPS using multiple polynomials over $F_{q'}$).

In this section we investigate for which λ, t, and n, the required number of polynomials over $F_{q'}$ exists without decreasing the security strength.

To construct a λ-degree bivariate polynomial, the set-up server needs to *randomly* select $\sigma = \binom{\lambda+2}{2}$ values from $F_{q'}$ [19]. Since the field $F_{q'}$ contains q' elements, each coefficient of a λ-degree bivariate polynomial over $F_{q'}$ can take up to q' values.

Therefore, the set-up server can choose up to ϕ different polynomials over $F_{q'}$ with σ coefficients:

$$\phi = (q')^{\sigma} \qquad \text{(Eq-6)}$$

Because the t polynomials included in a t-polynomial set are randomly taken from a set with ϕ elements, an attacker trying to fake a t-polynomial-set corresponding to any of the $\gamma = n^2 + n + 1$ t-polynomial-sets of the DPKPS has a success probability:

$$p_s = \gamma \times (1/\phi^t) \qquad \text{(Eq-7)}$$

4.5.6.1 Polynomials Security Condition

The number ϕ has to be large enough to guarantee that $p_s \to 0$ for target values of n, or, equivalently, it is practically impossible to fake any t-polynomial-set used in the DPKPS.

Let us assume that an attacker launches a brute-force attack to search a valid t-polynomial-set, similar to the one to break a symmetric key [52]. A symmetric key of 128 bits is an accepted value offering enough security strength for the following decades [51]. We want to achieve similar security strength for any generated t-polynomial-set, i.e. an attacker needs decades to find a t-polynomial-set belonging to the DPKPS.

Thus,

$$\gamma \times (1/\phi^t) \le 1/2^{128}$$

Let us now assume that γ can be approximated as 2^{Ψ}. Then,

$$(1/\phi^t) \le 1/2^{128+\Psi} \qquad \text{(Eq-8)}$$

Observe that fixed $t = t_0 = \lceil \lfloor \log q \rfloor / \lfloor \log q' \rfloor \rceil$, then ϕ is a function of λ. Consequently, the polynomials security condition sets a lower bound on λ:

$$1/(\phi(\lambda))^t \le 1/2^{128+\Psi} \Leftrightarrow \lambda \ge \lambda_{\min}, t = t_0, n \qquad \text{(Eq-9)}$$

4.5.7 FPP Existence

The DPKPS scheme solves the FPP existence problem of Çamtepe and Yener KPS [2][28] without the need to relax KPS connectivity by using the additional parameter *q'* yet enforcing the polynomials security condition (in section 4.5.6.1).

Assume we want, $N_{target} = 15501$, $m \leq 125$ and $\log q = 64$. Then, equation Eq-4 imposes $n < 125$. We can for instance choose $q' = 2^8 + 1$ ($\log q' = 8$ bits, optimal[25] for 8-bit CPUs). Then, equation Eq-2 imposes $n \geq 61$ to obtain $N \geq 15558 \geq N_{target}$. We can try choosing the next prime power, i.e. $n = 61$, and see if $\lambda = 1$ (imposed by equation Eq-4) satisfies the security condition. Note that $\gamma \approx 2^{12}$ and $\phi = (257)^3$ (by Eq-6). Thus, $1/\phi^t \approx 1/2^{192}$, which fulfils the security condition $1/\phi^t \leq 1/2^{140}$.

From the previous example we can derive that, for optimal value $q' = 2^8 + 1$ and minimum value $\lambda_{\min} = 1$, the value of *n* can be chosen up to $n \approx 2^{32}$ to fulfil the security condition. Thus, for practical values of *n*, the degree λ of the polynomials in the DPKPS can be chosen to be arbitrarily small without loss of security of polynomials.

In conclusion, for an attacker capturing nodes may result a cheaper approach to compromise t-polynomial-sets than forging t-polynomial-sets.

4.6 Practical Performance Evaluation

In this section we experimentally assess the performance of the DPKPS on WAHNs of low-power resource-restricted AquisGrain wireless nodes.

More specifically, we evaluate the DPKPS processes of t-Polynomial-Set Shares Discovery and Pairwise Key Establishment in terms of memory cost, bandwidth usage, computational cost as well as energy and time efficiency.

In the first two subsections, we introduce our test-bed and our implementation of the DPKPS. The rest of subsections present the performance results derived from our experiments.

4.6.1 Test-bed

Our test-bed consists of a WAHN formed with two AquisGrain wireless nodes. AquisGrain [54][55] is a node developed at Philips Research Laboratories Aachen for low-power wireless body sensor network (BSN) applications.

Figure 16 depicts the hardware components mounted on the main board of an AquisGrain. AquisGrain is provided with an ATmega128L low-power 8-bit micro-controller. It also includes a 128 Kbytes of flash memory, a 4 Kbytes EEPROM and a 4

[25] A value $q' = 2^{16} + 1$ can also be used while significantly improving scalability to the cost of a slight reduction of computational efficiency in the generation of keys.

Kbytes of SRAM. It is configured to work at a clock frequency of 7.4 MHz and it is sourced to a battery of 3 volt.

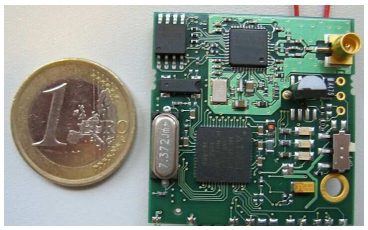


Figure 16. AquisGrain

AquisGrain uses the radio module CC2420 from Chipcon conforming the IEEE 802.15.4 standard [56]. The radio offers a maximum data rate of 250 Kbps in the 2.4 GHz frequency band and a typical range of 10 meters. The network layer complies the ZigBee^{TM} specification [57].

Because of its hardware and software configuration the AquisGrain wireless node is a good representative of the currently employed technology for wireless sensor nodes. A similar wireless sensor node (operating up to 8 MHz) is the MICAz mote developed by Crossbow.

4.6.2 Implementation of the DPKPS

The hardware characteristics of AquisGrain impose important restrictions in the implementation of the DPKPS.

First, the computations must be suitable for an 8-bit CPU without division operation. Consequently, we implemented polynomials with coefficients of 8 and 16 bits, since divisions[26] in F_{2^8+1} or $F_{2^{16}+1}$ can be computed without division operations [58]. We also implemented efficient algorithms to calculate modular operations in F_n.

Second, the DPKPS keying material and software stack must be of limited sizes. Consequently, we programmed efficient primitives to minimize the memory footprint.

We developed a security software module in AquisGrain nodes, which implements the processes of t-Polynomial-Set Shares Discovery and Pairwise Key Establishment of the DPKPS. That is, the security module enables two AquisGrain nodes to discover the identifier of their common t-polynomial-set share, to derive the respective points of evaluation and to respectively generate a pairwise key.

We developed a security protocol to discover t-polynomial-set shares and to establish a pairwise key. We also specified the packet format to communicate security modules in different AquisGrain nodes. The header of the security packet contains the following information:

[26] Required for modular operations.

1. *Security Identifier*. This field contains the identifier of the packet as belonging to the security module.
2. *Flow Control*. This field identifies the security packets within the security protocol.

The payload of the security packet contains one of the following contents:

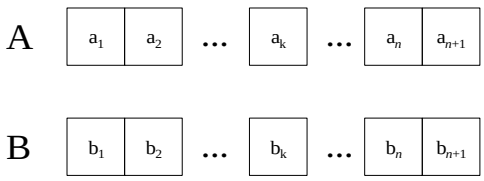
1. *Sensor ID*. This information is sent during the t-Polynomial-Set Shares Discovery phase.
2. *Acknowledgement*. This information is sent to acknowledge the proper reception of security packets.

The security packet is enclosed within the payload of a ZigBee™ network (NWK) layer packet.

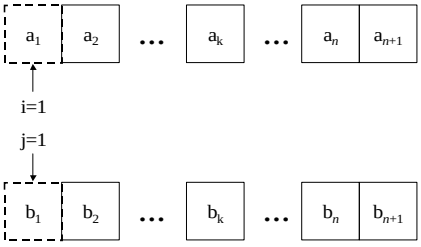
To identify the sensors, we implemented the optimized node ID of the DPKPS. This implementation enables us to evaluate the computational overhead added by generating FPPs and polynomial evaluation points on AquisGrain nodes.

The implemented generation of an FPP block automatically arranges the elements of the block in ascending order. To find the index *k* of the common t-polynomial-set shared by two AquisGrain nodes, the security module smartly searches through two sorted arrays containing the sorted elements of two FPP blocks.

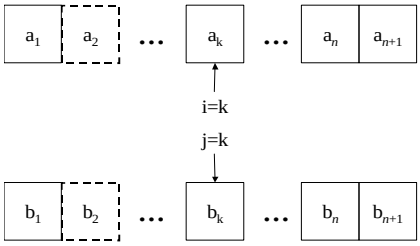
The process of searching works as follows. Consider the following two arrays A and B with elements sorted in ascending order (i.e. $a_i < a_{i+1}$ and $b_i < b_{i+1}$):



A and B have an element of same value at position *k*, which is unknown before the search algorithm runs. The search algorithm first indexes and compares the first position of both vectors:



If $a_i < b_j$ or $a_i > b_j$ a new comparison is needed and, then, the index *i* or *j* is accordingly augmented a unit. This operation is repeated at most 2*n* times till the element $a_k = b_k$ is found:



The keying material, including the optimized node ID and t-polynomial-set share coefficients, is encoded in a vector. The first position of the keying material vector contains the ID and the rest of positions are sequentially occupied by the $\lambda+1$ coefficients of each of the $n+1$ t-polynomial-set shares associated to that node ID. The keying material vector is loaded in the memory of a sensor and is eventually accessed by the security module on execution time.

Note that because F_{2^8+1} contains 2^8+1 elements, in order to codify t-polynomial-set share coefficients 9 bits are theoretically needed. Since AquisGrain memory spaces are multiples of 8 bits, a naïve implementation would require variables of 16 bits to codify each coefficient (this, in turn, would roughly double the memory requirements for keying material in respect to the theoretical analysis in Section 4.5.3).

Observe however that the 9-th bit is set to 1 in just one element of the field, e.g. for the coefficient with value 2^8+1. In the rest of coefficients, namely 1 to 2^8, the value of the 9-th bit is to be set to 0.

Consequently, we designed and applied a codification method in which all the coefficients are encoded in just 8-bit variables. The coefficients with value 1 and 2^8+1 are both encoded with the 8-bit binary code 00000000. To differentiate between the coefficient 1 and 2^8+1 we used the bits of additional 8-bit variables (hereafter denoted as AVs), where each bit encodes the 9-th bit of a coefficient with binary code 00000000. The order of bits within one of these AV variables corresponds to the occurrence of the code 00000000 within the keying material vector.

Each AV variable is used to encode up to 8 occurrences of 00000000. Because in average a coefficient with value 1 or 2^8+1 occurs just once out of 2^8+1 stored coefficients, by using this method the memory usage in practice on nodes is m. Therefore, approaching its theoretical value.

For our implementation with coefficients over $F_{2^{16}+1}$ we applied the same codification concept. In this case, the value of the coefficients was encoded in 16-bit variables. The length of the AV variables, used to codify the 17-th bit of 1 and $2^{17}+1$, was just 8 bits.

We also developed the Keying Material Generator, a JAVA application that automatically generates the desired number of keying material vectors for user-introduced values of λ, n, log$q$ and log$q$'. This program implements the functionality of the DPKPS set-up server.

The length of the generated pairwise keys is 64 bits, both in our experiments with 8-bit and 16-bit polynomial coefficients.

4.6.3 Memory Requirements

The memory needs on an AquisGrain are divided in requirements of static and dynamic memory. Static memory is required to store the security software module and keying material. Dynamic memory is storage needed to store runtime variables during execution time.

Table 3 quantifies in bytes the footprint of our implementation. The security software module occupies just a 1% of the available flash memory and runtime variables occupy just a 0.32% of the available SRAM.

Naturally, the memory occupied by the keying material depends on the selected values for λ and n. For instance, for $n = 11$ and $\lambda = 9$ the keying material occupies nearly 1 Kbyte, which results in just a 0.75% of the total static memory capacity (an acceptable value in AquisGrain nodes).

Additionally, each AV variable adds one byte. In our experiments we found that the memory overhead from AV variables can be neglected. For instance, the expected number of bytes added from AV variables needed for 1 Kbytes of keying material is 1 byte (in the 8-bit coefficient case) and 0 bytes (in the 16-bit coefficient case).

Table 3. Memory Cost

Security Item	Memory Type	Memory Cost (bytes)
Keying Material	EEPROM	$8(\lambda+1)(n+1)$
Security Module	FLASH	1416
Runtime Variables	SRAM	13

In conclusion, from the memory overhead point of view the implementation of the DPKPS in AquisGrain nodes with 1 Kbytes keying material leaves 98% of the available static memory and 99.68% of the available dynamic memory for other data and software applications.

4.6.4 Bandwidth Usage

The bandwidth usage is calculated as the number of bytes exchanged by two AquisGrain nodes to establish a pairwise key. This number is calculated adding the length of two security packets with the overhead added by the ZigBee™ layers.

4.6.4.1 Security Packet Size

In our implementation we restricted the amount of keying material to 1 Kbytes. Consequently, the value of n is at most 61.

The header of the security packet is 2 bytes long. The first byte is used to identify the security module (among other applications running on the AquisGrain). The second byte is used for flow control.

The payload of the packet includes the optimized sensor ID (refer to Section 4.4.5 for related theory) and is at most 4 bytes long. The first 12 bits are used to encode an FPP block identifier. In such case, the number of FPP blocks to codify is 3783, which can be

codified with 12 bits. The second 11 bits are used to encode the turn i_p in the distribution of t-polynomial-set shares to a sensor. The number of turns is given by the formula $\frac{q'-1}{n+1}$, which is maximized for $q' = 2^{16}+1$. Then, for $n = 61$, the maximum number of turns is 1057, which can be codified in 11 bits. The third 6 bits are used to encode an element of the Latin square L_i from which B_i is derived. Since the number of elements of L_i is $n = 61$, these elements can be encoded in 6 bits. The last 3 bits of the payload are left unoccupied.

In conclusion, the aggregated maximum length of a security packet is 6 bytes.

4.6.4.2 ZigBee/802.15.4 Headers Overhead

In our implementation, two security modules communicate at the ZigBee™ network (NWK) layer, i.e. a security packet is enclosed within the payload of the NWK packet. Consequently, the headers of ZigBee™ NWK and lower layers must be considered to evaluate the bandwidth usage of our implementation.

ZigBee/802.15.4 specifies a physical layer header of 6 bytes. The size of the MAC layer header varies from 5 to 13 bytes.

Additionally, ZigBee/802.15.4 defines an optional MAC security header of 4 to 21 bytes. The NWK header size varies from 2 to 13 bytes.

In our experiments, we set the MAC and NWK headers to 23 and 6 bytes, respectively. The MAC address of each AquisGrain is a fixed value of either 2 or 8 bytes long. Since the maximum number of different AquisGrain nodes that can be configured using our implementation is 4064306, to cover this number we decided to set MAC addresses of 8 bytes.

The NWK address of each AquisGrain is a dynamic value of 2 bytes long, which is assigned to each AquisGrain in joining the WSN. The rest 4 bytes include flow control and error correction values.

In conclusion, ZigBee™/802.15.4 headers add 35 additional bytes.

4.6.5 Computational Cost

The application of the DPKPS to establish pairwise keys exhibits nice features for WAHN security. The trade off is additional computational overhead on AquisGrain nodes.

In the following subsections we quantify and assess the computational overhead of DPKPS functions demanding CPU computations. The results in the following subsections are presented in terms of CPU cycles of the ATmega128L.

4.6.5.1 Generation of t-Polynomial-Set Identifiers (from Optimized Node Identifier)

To generate the n +1 identifiers of the t-polynomial-set shares stored in the AquisGrain, the security module computes from the optimized node ID the n +1 elements of an FPP block.

A block B_i is generated using three different methods for the cases $1 \le i \le 2n$, $2n < i \le n^2 + n$ or $i = n^2 + n + 1$ respectively (see Sections 4.4.5.2 and 4.4.5.3).

Figure 17 depicts the computational cost of generating a block with the first two methods. In both cases, the cost is not significant for practical values of n. The cost of the third method is still more negligible and, thus, not depicted.

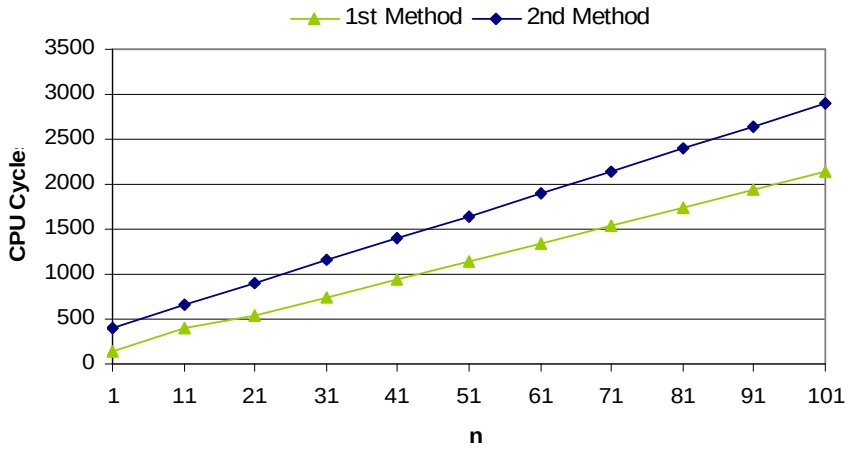


Figure 17. Computational cost of FPP block generation

4.6.5.2 Search of Common t-Polynomial-Set

Figure 18 depicts the computational cost for the best and worst search cases to find a common t-polynomial-set.

The best case happens when the common element occupies the first position in both vectors containing t-polynomial-set identifiers. In such a case, the search algorithm finds the common element in the first round.

Similarly, the worst case counts the case when the common element occupies the last position in both vectors. Consequently, the search algorithm finds the common element in the last round, which involves $2(n+1)$ comparisons.

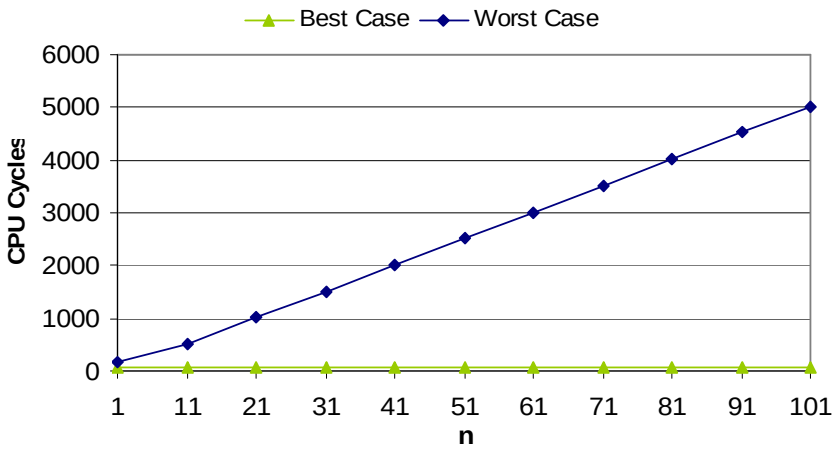


Figure 18. Computational cost of searching the index of a common t-polynomial-set

4.6.5.3 Calculation of Point of Evaluation (from Optimized Node Identifier)

Depending on the range of the index *k* of a t-polynomial-set and the index *i* of block B_i, the evaluation point can be calculated using three different methods (see Section 4.4.5.4).

The implemented algorithm to calculate evaluation points uses 16-bit variables and it is, then, applicable for both the cases $N' = 2^8$ and $N' = 2^{16}$, and for *n* up to 2^{16}.

The first and the second methods require just 300 and 480 CPU cycles, respectively. The computational cost of the last method is negligible.

4.6.5.4 Generation of a Pairwise Key

Figure 19 shows the computational cost of generating a 64-bit pairwise key with AquisGrain for increasing value of λ when $q' = 2^8 + 1$ or $q' = 2^{16} + 1$. To provide an intuition of the depicted magnitudes, we compare with the cost of encrypting a 64-bit message using RC5 (RC5 encryption is extremely computationally efficient).

Figure 19 shows that the cost of generating a pairwise key is equivalent to encrypting a 64bit message with RC5-MAC when $\lambda = 5$ (case $q' = 2^8 + 1$) and $\lambda = 4$ (case $q' = 2^{16} + 1$).

Note that generating a 64-bit key takes approximately 4 times the cost of computing a 16-bit partial key using polynomials over $F_{2^{16}+1}$. Accordingly, it takes approximately 8 times the cost of computing an 8-bit partial key using polynomials over F_{2^8+1}.

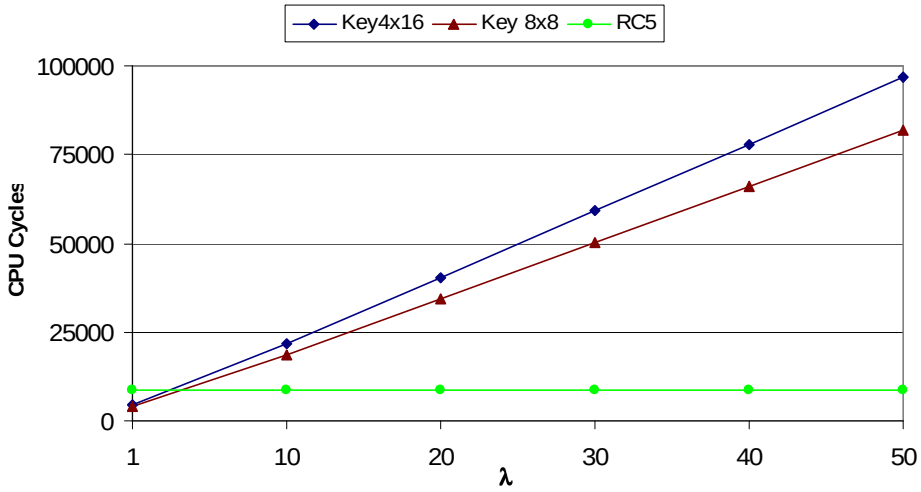


Figure 19. Computational cost of generating a pairwise key of 64 bits

As Figure 19 shows, for λ in the range 1 to 100, a reduction of the degree of t-polynomial-sets over F_q, can improve the computational efficiency on nodes up to 2 orders of magnitude.In the DPKPS (as we show in sections 4.5.2 and 4.7) the value of λ can be chosen to be of small value while keeping the resiliency of the KPS a constant α.

4.6.5.5 Aggregated Computational Cost

Figure 20 and Figure 21 depict the total computational cost of calculating a 64-bit pairwise key resulting from t-polynomial-sets over F_{2^8+1} and $F_{2^{16}+1}$, respectively. Both pictures demonstrate that the cost of generating FPP blocks and evaluation points (depending on *n*) is negligible with the cost of generating a key from a t-polynomial-set (depending on λ).

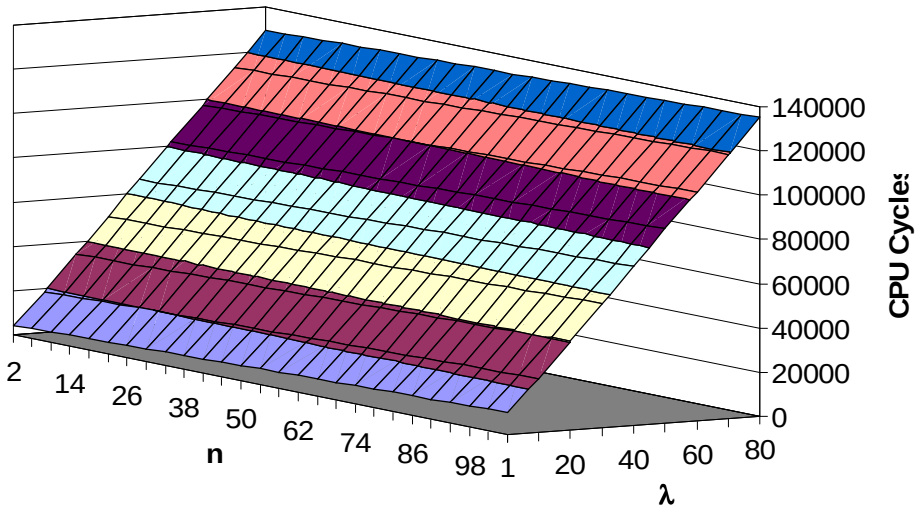


Figure 20. Aggregated computational cost of generating an 8x8-bit pairwise key

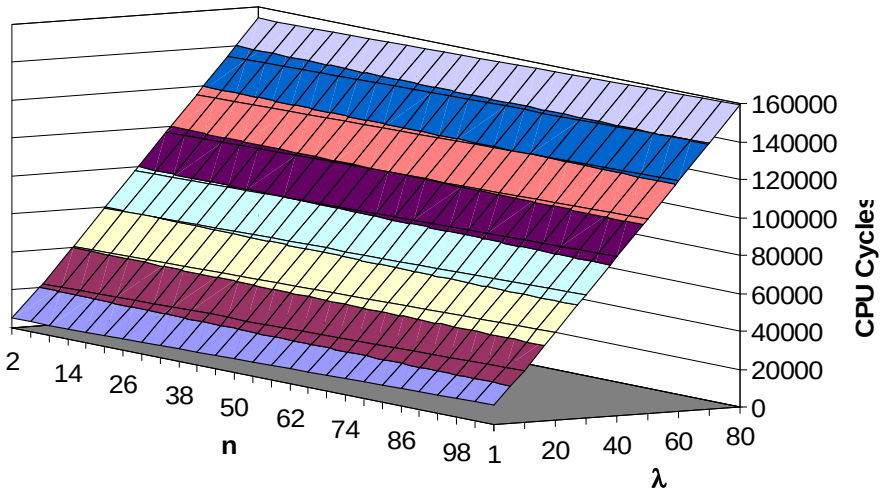


Figure 21. Aggregated computational cost of generating a 4x16-bit pairwise key

The conversion from CPU cycles *C* to time *T* is given by the following formula:

$$\mathbf{T} = \text{C (cycles)} / (\text{Clock\_Frequency (cycles/s)} = \mathbf{C \times 10^{-3}/7.4\ ms.}$$

Considering the clock frequency of 7.4 MHz, Figure 22 shows the time needed by an AquisGrain node to compute a pairwise key 4x16-bit.

For $n = 11$ and $\lambda = 9$ the time of computation is lower than 1 ms. For $n = 61$ and $\lambda = 1$, the time of computation is slightly higher than 1 ms. WSN applications allowing delays up to 22 ms may implement DPKPS with values up to $n = 103$ and $\lambda = 80$ (naturally, if the memory of the sensors can allocate the drastic corresponding amount of keying material).

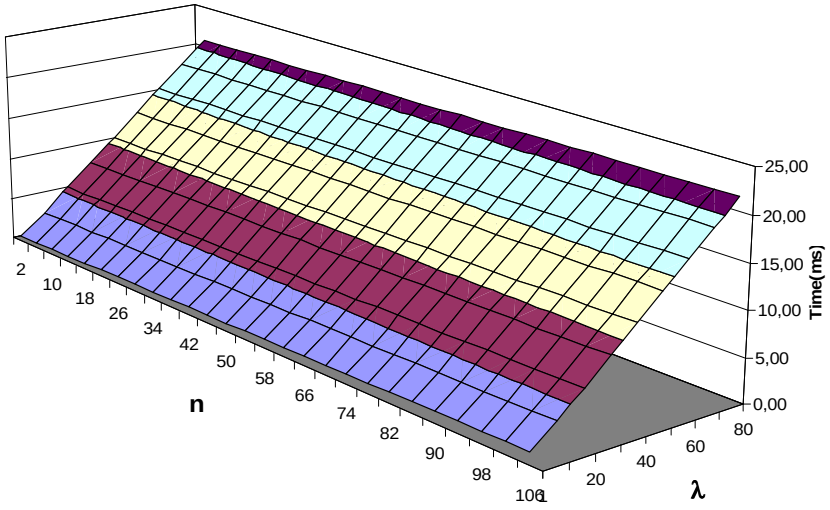


Figure 22. Aggregated time cost to generate a 4x16-bit pairwise key

4.6.6 Energy Efficiency

The energy consumption of the Atmega128L depends mainly on the operating voltage, clock frequency and temperature.

At the frequency of 7.4 MHz, power supply of 3 V and temperature of 25 °C, the current consumption of the Atmega128L is approximately 8 mA (see Figure 23).

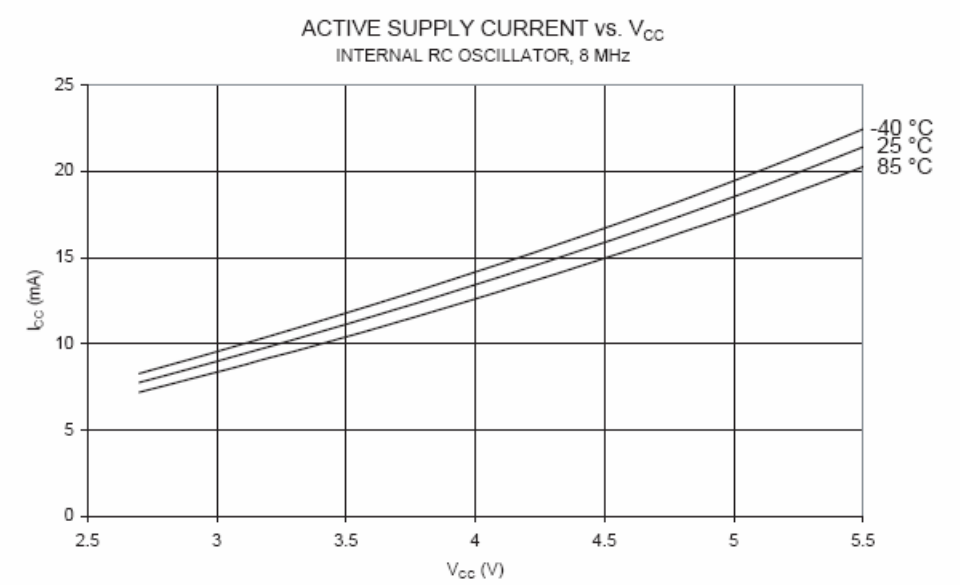


Figure 23. ATmega128L Current Consumption at 8 MHz (Source [59])

The conversion from CPU cycles *C* to energy *E* is given by the following formula:

E = C (cycles) x I (mA)/(Clock_Frequency (cycles/s) *3600(s/h)) = **C x3x10^{-10} mAh**

The current consumption of the CC2420 radio module is 19.7 mA in reception (Rx) and 17.4 mA transmitting (Tx) at the maximum power of 0 dBm [60]. Note that the CC2420 consumes less power in Rx mode than in Tx mode, since the decoding operations in Rx are more complex than the encoding operations in Tx.

The conversion from bits *B* transmitted/received in/from to energy *E* is given by the following formula:

E = B (bits) x I (mA)/(Data Rate (bits/s) x 3600 (s/h))

, where I = 19.7 or 17.4 mA (Rx or Tx, respectively)

Consider now that two AquisGrain nodes establish a pairwise key. Consider the AquisGrain node powered with an inexpensive 3-V coin cell battery of minumum capacity, just 30 mAh.

Let us analyze thecase when $n = 11$ and $\lambda = 9$ (Recall a node would require less than 1 kbytes for storing keying material).

The energy consumed by each AquisGrain is the aggregate of the energy invested in exchanging sensor IDs with the energy used during the computation of a pairwise key. Since at the physical layer the AquisGrain node sends and receives two packets of 41 bytes each (refer to Section 4.6.4), the energy spent in communication is 13.52 nAh. The energy spent in generating the key is only 5.75 nAh.

By investing *only* a 1% of its battery resource, the AquisGrain node can generate 15568 different pairwise keys. Intuitively, by investing only a 1% of its battery the AquisGrain can generate a new pairwise key every hour during almost two years!

In a mobile scenario, this allows a node to secure 24 new encounter with other nodes every day during nearly two years. Similarly, by dedicating a 10% of battery for security, a node can secure 240 encounters every day for the same time period.

In conclusion, the DPKPS results drastically power efficient for pratical WAHN applications.

4.7 Comparison with Deterministic KPS Schemes

The DPKPS presents advantageous properties when compared with deterministic KPSs.

In Table 4 and Figure 24 we compare the DPKPS scheme with previous deterministic KPSs [19][28] that also enable direct key establishment in terms of resiliency against node captures, scalability and revocability [30].

For comparison, the resiliency of a deterministic KPS is calculated as the minimum number of nodes that a smart attacker needs to capture to compromise all the pre-distributed keys. The scalability of two deterministic KPSs can be compared in terms of the maximum number *N* of nodes that can be accommodated for the same resiliency level. A scheme with revocation enables the dynamic removal of a misbehaving or compromised node from the system.

Because the DPKPS enables a node to be uniquely identified and authenticated, it also enables revocation of nodes (see column "**R**" of Table 4). Column "***m***" of Table 4 shows the parameters to evaluate the memory *m* required by each KPS on a node. In the trivial (see Section 2.5.4), Blundo [19], combinatorial based on FPP [28] and DPKPS KPSs the resiliency level "**α**" is limited by the memory size *m*. In the trivial and combinatorial KPSs, the scalability "**N**" is also restricted by *m*. The scalability of Blundo's[27] KPS is restricted by the finite field prime order *q*' where the polynomials are chosen from. The scalability of DPKPS[5] is determined by both *q*' and a fraction of *m* (see Figure 24, for $n = m/2-1$, $m=125$, i.e. $\lambda = \lambda_{\min} = 1$, $q' = 2^8 + 1$. For simplicity's sake, Figure 24 does not restrict *n* to prime powers).

Table 4. KPS comparison

KPS	R	m	α	N
Network Key	No	1	1	Unlimited
Trivial	Yes	N-1	N-1	N
Blundo	Yes	$\lambda^B + 1$	$\lambda^B + 1$	q' -1
Comb-FPP	No	$\leq n^{FPP} + 1$	$n^{FPP} + 1$	$(n^{FPP})^2+n^{FPP}+1$
DPKPS	Yes	$(\lambda+1)(n+1)$	$(\lambda+1)(n+1)$	$(q' -1)(n^2+n+1)/(n+1)$

[27] We assume an implementation taking polynomials over F*q*' (for optimal computational efficiency) and $\lambda^B + 1 = m$.

For the same resiliency level $\alpha \approx m \leq 125$ and $q' = 2^8 + 1$, the DPKPS can accommodate more nodes than the rest of deterministic KPSs (See Figure 24).

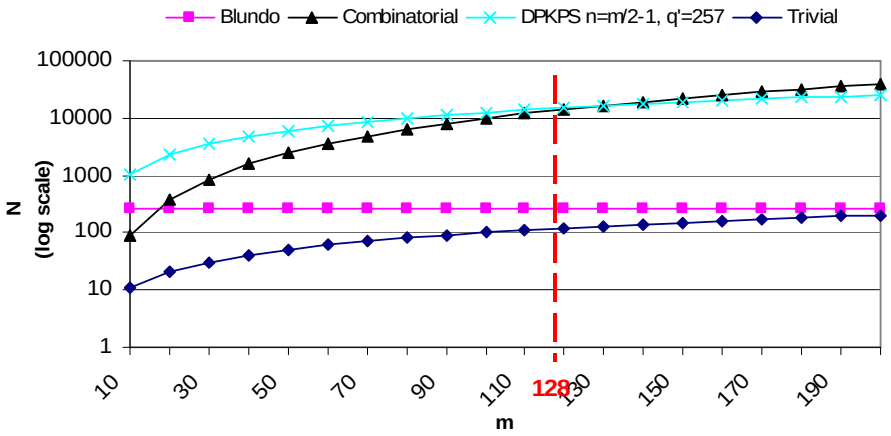


Figure 24. Scalability of KPSs ($q' = 2^8 + 1$)

Figure 24 shows that for $m = N'/2 = 128$, both DPKPS and the combinatorial KPS based on FPP [28] can accommodate the same number of nodes whereas for $m \geq N'/2$ combinatorial KPS can accommodate more nodes.

In general, for $n = m/(\lambda + 1) - 1$ and q', both KPSs can accommodate the same number of nodes when a memory of size $m \approx N'/(\lambda + 1)$ is considered in nodes. Consequently, choosing a larger value of q' (i.e. N'), the DPKPS has a greater scalability for practical values of m. (see Figure 25).

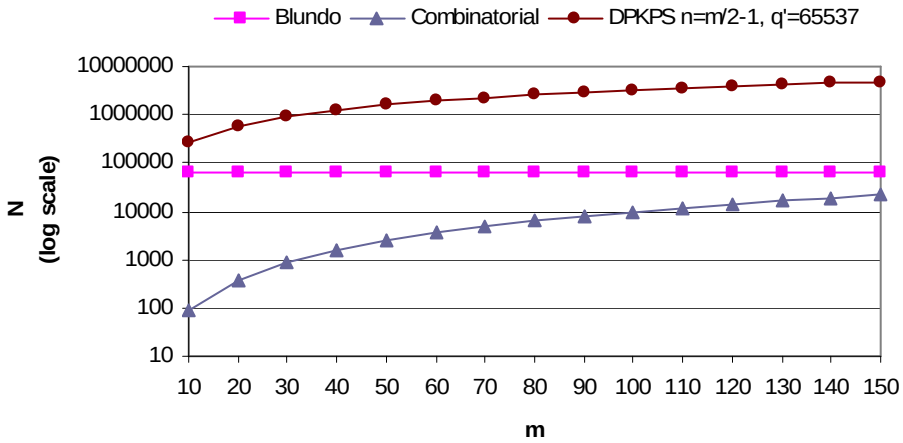


Figure 25. Scalability of KPSs ($q' = 2^{16} + 1$)

Since in the DPKPS and Blundo KPS nodes generate pairwise keys dynamically from polynomials, the computational cost of establishing a key is slightly higher than in any of the other deterministic KPSs discussed in this section.

To establish a common key, the DPKPS and Blundo KPS mainly require modular operations in $F_{q'}$ while the rest of deterministic KPSs require searching the key in the

memory of the node. As we show in the practical performance evaluation (section 4.6.5.1), generating a pairwise key in the DPKPS is extremely efficient in strictly constrained CPUs.

Assume that using Blundo's KPS we pre-distribute *N*' shares of a λ^B-degree t-polynomial-set to exactly *N*' nodes, where $\lambda^B + 1 \leq m$. In the following, let $q' = 2^8 + 1$ and $m = 125$. For comparison, now assume that we want to accommodate *N*' nodes using the DPKPS.

In such a case, we can choose $n^2 + n + 1 = N'$, i.e. $n = 2^4$ (maximum supportable network size 4096), and assign each node to a different FPP block. The degree of the $n^2 + n + 1$ t-polynomial-sets is exactly $\lambda \equiv m/(n+1) - 1 \equiv 6$.

Alternatively, we can choose $n = 2$ and $\lambda = 40$ to accommodate the *N*' nodes (but the maximum supportable network size is only 597 nodes).

Using the previous parameters, Figure 26 compares the resiliency of Blundo's KPS and the DPKPS under wise and naïve attacker models.

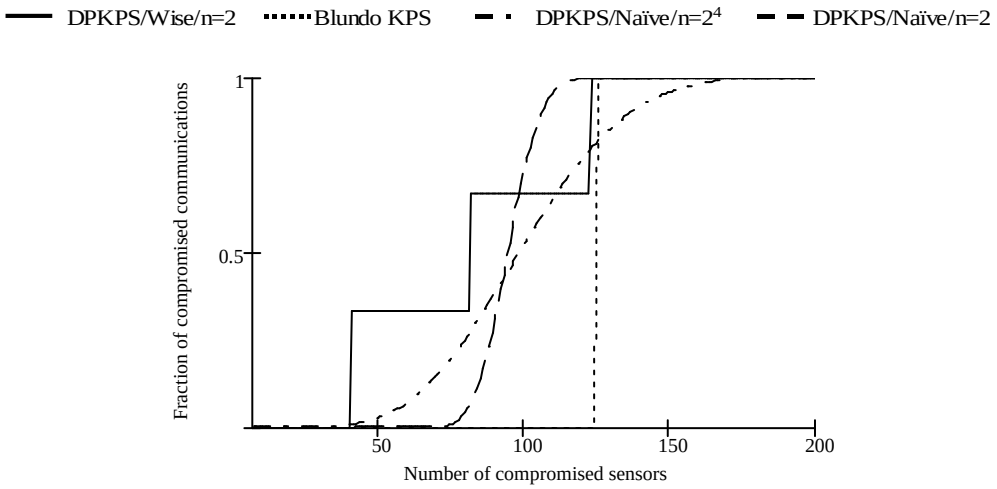


Figure 26. DPKPS vs. Blundo's KPS resiliency

Figure 26 shows that by randomly capturing *m* nodes the attacker compromises 100% of the communications if Blundo KPS is used, whereas, if the DPKPS is used to accommodate exactly *N*' nodes, the fraction of compromised communications is 81,7%.

Observe that, under the naïve attacker model, for $\lambda = 6$ and $\lambda = 40$ perfect resiliency of the DPKPS is broken when approximately 40 and 80 nodes are captured, respectively. Observe that for equivalent perfect resiliency, a Blundo KPS with $\lambda^B = 39$ or $\lambda^B = 80$ demands respectively 6 and 2 times more computational effort than the DPKPS (see Figure 19). Additionally, the DPKPS can accommodate much more nodes.

4.8 Comparison with Random KPS Schemes

The DPKPS presents advantageous properties when compared with random KPSs [20]-[24][26]. Firstly, the DPKPS enables direct pairwise key establishment. Then, improving random KPSs, any pair of nodes can establish a pairwise key with

independence of the WAHN size and the density of a node's wireless neighborhood. The pairwise key can be used for node authentication and for protection of communications

Secondly, since third-party nodes are not used for key establishment in the DPKPS, mediated key establishment using compromised nodes does not pose a security risk as in random KPSs.

Thirdly, since DPKPS key establishment just involves the two nodes establishing the key, without the need to find and communicate with one or more trusted third-party nodes, the communication and computational overhead is also reduced in comparison to random KPSs.

A smart attacker, who selectively captures nodes, is a realistic attacker model for key pre-distribution systems [32], especially in high security-sensitive applications (such as military).

Let us first analyze and compare with *key pool*-based random KPS [20][21][24][26] under the smart attacker model. Assuming that the attacker captures nodes carrying different keys in their key rings, the resiliency of random key pool-based KPSs is reduced to $\alpha^{smart\_KeyPool} \approx |S|/k$, where $|S|$ denotes the number of keys in the initial key pool and *k* the number of keys in a node's key ring (note that the maximum value of *k* is *m*). Recall that, in this case, the resiliency of the keys generated with the DPKPS is $\alpha^{smart\_DPKPS} = (\lambda+1)(n+1) = m$.

In random key pool-based KPSs, the probability p that two nodes share at least one key is a function of $|S|$ and *k*, and can be calculated after Eschenauer and Gligor [20]:

$$p = 1 - \frac{((|S|-k)!)^2}{(|S|-2k)!|S|!}$$

The formula of *p* shows that if we want two arbitrary nodes to be able to establish a common key with $p=1$ (as in the DPKPS), then $|S|/k \to 1$. In other words, the random KPS tends to a system key KPS, and then $\alpha^{smart\_KeyPool} \to 1 << \alpha^{smart\_DPKPS}$. Alternatively, we can set $\alpha^{smart\_KeyPool} \approx |S|/k > \alpha^{smart\_DPKPS} = m$. In this case, the resiliency of a random KPS against the smart attacker is higher, but then *p* is reduced.

Now we analyze and compare with polynomial (or similarly with Blom key-space) pool-based random KPS [23][22]. In this random KPS, a node carries *s*' distinct polynomial shares and the initial pool contains *s* distinct polynomials. The resiliency of the KPS against the smart attacker is $\alpha^{smart\_PolyPool} = s(\lambda+1)$ while the memory used in nodes is $m = s'(\lambda+1)$. The probability p that two nodes can directly establish a pairwise key is a function of *s* and *s*', and can be calculated after Liu and Ning [23]:

$$p = 1 - \prod_{i=0}^{s'-1} \frac{s-s'-i}{s-i}$$

The formula of p shows that if we want two arbitrary nodes to be able to establish a common key with $p=1$ (as in the DPKPS), then $s' \to s$. In other words, the random KPS tends to a Blundo KPS (where each node receives s' shares of $s=s'$ distinct polynomials). Consequently, $\alpha^{smart\_PolyPool} \to s'(\lambda+1)=m$, which coincides with $\alpha^{smart\_DPKPS}=(\lambda+1)(n+1)=m$. However, in such a case, the DPKPS can accommodate more nodes than the random KPS, i.e. $N^{PolyPool}=q'-1$ while $N^{DPKPS} \approx n(q'-1)$. Alternatively, we can let $\alpha^{smart\_PolyPool}=s(\lambda+1) > \alpha^{smart\_DPKPS}=m$. In this case, the resiliency of a random KPS against the smart attacker is higher, but then p is reduced.

4.9 Summary and Conclusions

In this chapter, we have presented the deterministic pairwise key pre-distribution scheme (DPKPS). The DPKPS enables security in WAHNs of strictly resource-constrained nodes. In an initial configuration phase a set-up trusted server pre-distributes keying material to WAHN nodes. Afterwards, any pair of nodes initialized with the DPKPS can directly establish a pairwise key in a extremely efficient manner. This property makes it equally suitable for full-connected or partially-connected WAHNs of arbitrary membership size.

The DPKPS keying material consists of a number of optimized Blundo's polynomials. To optimize scalability, security robustness and energy efficiency, we use a combinatorial selection method to distribute a number of shares of different polynomials to each WAHN node.

We have shown that for similar network resiliency level and memory usage, the DPKPS can accommodate a greater value of nodes than other deterministic KPS approaches without paying off a substantial energy overhead. Because the DPKPS enables direct key establishment between any pair of nodes of a system, it can be applied to WAHNs of static or mobile nodes and of arbitrary node population size and density. Random KPSs relax connectivity properties for network scalability or resiliency and, thus, they perform well only in static WAHNs.

We have also derived a lower bound on the degree λ of the polynomials over $F_{q'}$ to guarantee their security while simultaneously optimizing their computational performance on constrained nodes. This result is applicable to other existing polynomial-based KPSs and, particularly, to the DPKPS.

The DPKPS can be applied to provide security in networks of resource-restricted AquisGrain nodes. One application of special interest is the protection of medical BSNs, which are used for patient vital sign monitoring. In this context, the DPKPS enables the pre-configuration of trust relationships among potential medical wireless devices before they are actually used at the BSN. This setting further enables a clinician to attach or to associate two or more such devices and automatically create a patient BSN, without the need to actively configure security for that BSN. Because DPKPS-based security is plug&play and user transparent, the clinician can exclusively focus on diagnosing the patient.

Further work includes investigating approaches to increase the resiliency of the DPKPS without relaxing connectivity properties or increasing memory and energy cost. One

such approach may explore mechanisms to minimize correlation between keying material stored in nodes associated to the same FPP block.

A key management infrastructure can be designed on the basis of the DPKPS. The key management infrastructure may be further extended with appropriate keying material update and revocation mechanisms. Because the ad hoc disconnected nature of MSNs, solving this functionality in this context results a complex problem [3].

Finally, the design of access control mechanisms, including role, group and user access control, in WAHNs of resource-constrained nodes is also a line of research yet to go. The DPKPS serves the basic identification and keying service on top of which access control services may in the future be defined.

5 Demonstrator for Medical Body Sensor Networks

As proof of concept, we integrated our implementation of the DPKPS (see Section 4.6.2) in a demonstrator for patient vital sign monitoring enabled by AquisGrain nodes. We have also shown the functionality of the DPKPS in typical patient monitoring scenarios.

This chapter is organized as follows. In Section 5.1 we describe the context of medical monitoring with body sensor networks (BSN). Section 5.2 discusses models to employ the DPKPS for security in hospital BSNs. The demonstrator system is technically detailed in section 5.3. In section 5.4 we assess the feasibility of the DPKPS for typical medical BSN scenarios. Section 5.5 briefly summarizes and concludes the chapter.

5.1 Medical Monitoring with Body Sensor Networks

Continuous monitoring of vital signs (such as electrocardiogram (ECG), heart rate (HR) and pulse oximetry (SpO2)) is an essential element for the treatment of patients.

For the physiological measurement of vital parameters, vital sign sensors need to be attached to the patient body. In the current status, sensors are typically wired to a bedside monitor for continuous display of patient's vital sign waveforms, numerics and alarms. This set-up ties patients to their beds, which is inconvenient for them. It also complicates work for caregivers since the wires hinder access to patients. In addition, the number and type of sensor nodes attached to the patient is strictly limited by the number and type of input wire ports at the display device.

To solve the previous limitations and reduce the costs of caregiving in the future hospital, the wireless BSN will enable the acquisition, processing, storage and, when possible and/or needed, the transmission of the patient vital signs to a display device (see Figure 27). BSN nodes are inexpensive wireless sensor nodes with restricted data-collection, -processing and -storage as well as radio transmission capabilities.

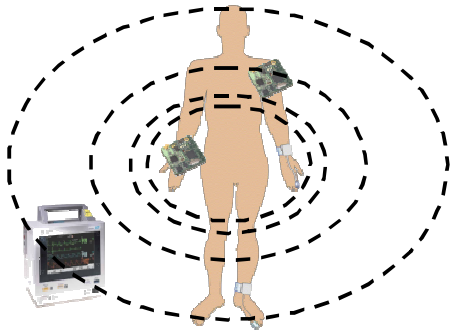


Figure 27. A body sensor network

The sensor nodes are appropriately pre-configured to enable easy and quick set-up of a BSN. The clinicians will form a BSN just by placing (and subsequently adding or removing) wireless sensor nodes at the patient’s body, without any obtrusive configuration effort. These nodes are randomly selected from a pool with a large number of other nodes. Moreover, as long as batteries last, these nodes can be re-used

for subsequent patients without requiring software updates, i.e. they can be detached from an old patient and associated to a new patient.

The patient will be free to move everywhere in the hospital. Eventually, a clinician will come at the patient's vicinity (e.g. during the clinician round) and monitor the patient on his/her PDA (or a PBM). Note that any clinician at the hospital may appear on the patient vicinity and monitor the patient anytime.

Security is of particular concern in BSN settings. BSNs are formed in public (and hostile[28]) areas where wireless communication may be monitored (see Figure 28). Additionally, sensor nodes may be subject to capture and manipulation by an adversary. Particularly, in hospitals access to BSNs (this comprises device association to a BSN and data access) must be restricted to authorized hospital entities (including medical sensors, devices and clinicians).

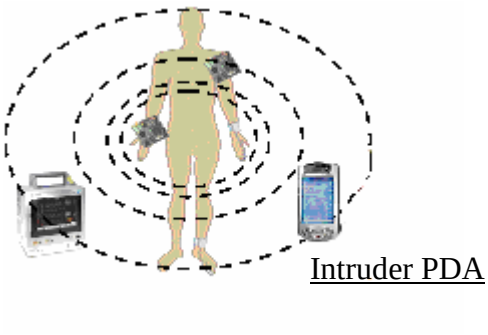


Figure 28. The need for security in BSNs

Because of the system limitations, operational demands and security risks discussed in former paragraphs, medical BSNs require low-power low-cost effective security services of authentication, communication confidentiality and integrity as enabled by the DPKPS.

5.2 DPKPS Use Models in the Hospital

There are two options to set DPKPS-enabled security in a hospital, which differ in the entity administering security.

In the first (and recommended) option, the Biomedical Engineer (BioEng) - an engineer being responsible for all the medical equipment in the hospital - configures security in the new wireless medical equipment after purchasing it.

Initially, the software of the medical equipment includes a security module (similar to that of Section 4.6.2) and no keying material. Consequently, the hospital needs to buy a Keying Material Generator to configure security on wireless medical equipment. The Keying Material Generator is to be installed in a set-up server, e.g. a PC used for set-up purposes.

[28] e.g. in military applications.

To configure security, the BioEng shall first estimate the number of medical devices to be set-up with DPKPS keying material as well as the potential number of medical devices to be purchased in successive orders. For instance, assume that the total number of medical devices purchased and to be purchased is 100000 and that the number of currently purchased devices is 1000. Then, by using the Keying Material Generator, the BioEng sets the parameters of the DPKPS to enable the generation of keying material to configure at least 100000 devices. In this point in time he just generates and loads keying material vectors to the first 1000 purchased devices.

The keying material shall be loaded under strict security measures to avoid that an attacker may learn the keying material. For instance, if keying material is downloaded using in-band mechanisms, then the wireless transmission shall be confined to a highly restricted physical area.

After security configuration, the BioEng deploys the new 1000 wireless medical devices in the hospital, i.e. he distributes them among different hospital departments and clinicians. The clinicians can now form secure BSNs with any combination of the 1000 devices.

Imagine now that the BioEng orders 999 additional medical wireless devices. Because the Keying Material Generator stores the parameters of the initial security configuration, the BioEng can configure in the new 999 devices DPKPS keying material that is different yet interoperable with keying material loaded in the first 1000 devices. Consequently, the new 999 and the old 1000 can now establish secure BSNs formed with any combination of the 1999 devices.

The second option is targeted at hospitals without a BioEng (we discourage it in hospitals with a BioEng). The medical equipment provider (MEP) assumes the responsibilities of a security administrator for its customers. For each customer, the MEP estimates the total number of devices to be delivered (possibly, also in successive orders) and installs DPKPS keying material during production. To avoid mixing up security domains of different hospitals, the keying material installed in devices delivered to two different customers shall be uncorrelated.

Note that with this second option the MEP can offer a complete plug&play and secure solution, i.e. new medical equipment can directly be deployed in the hospital without the need for security configuration efforts from hospital personnel. The wireless medical devices can establish secure BSNs.

From the hospital perspective, the disadvantages of this second option lie first on the need to unconditionally trust the MEP as a security administrator and, second, on the impossibility to autonomously manage the DPKPS keying material of its medical devices. For instance, to update DPKPS keying material from the medical devices, the hospital needs to hire the MEP as a security administrator.

5.3 Demonstrator System

We run the JAVA-based Keying Material Generator to generate DPKPS keying material in a normal computer, which simulates the DPKPS set-up server.

Furthermore, we integrated our implementation of the DPKPS in a demonstrator for patient vital sign monitoring developed for the "Cableless Patient" project at Philips Research Laboratories (Aachen).

Figure 29 depicts the protocol stack used for the security demonstrator. The DPKPS is placed in each AquisGrain on top of the ZigBee™ NWK layer. On top of the DPKPS we implemented a simple authentication protocol. The top of the stack includes ECG, SpO2 and display applications. The ECG and the SpO2 applications generate simulated ECG and SpO2 data, respectively. Only after successful authentication, ECG, SpO2 and vital sign display applications are allowed to intercommunicate.

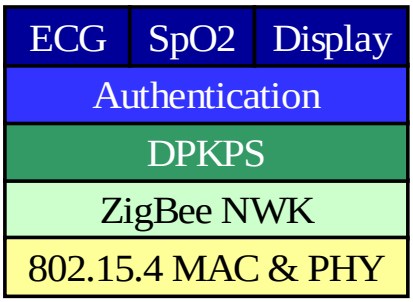


Figure 29. Protocol stack used for demonstrator

Two laptops, which simulate either PBMs or clinician PDAs, run the vital sign display application. The laptop is connected via the serial port interface to an AquisGrain node. Two additional AquisGrain nodes run the ECG application. The third AquisGrain runs the SpO2 application.

Figure 30 shows a possible configuration of the demonstrator with a laptop and one ECG AquisGrain and the SpO2 AquisGrain.

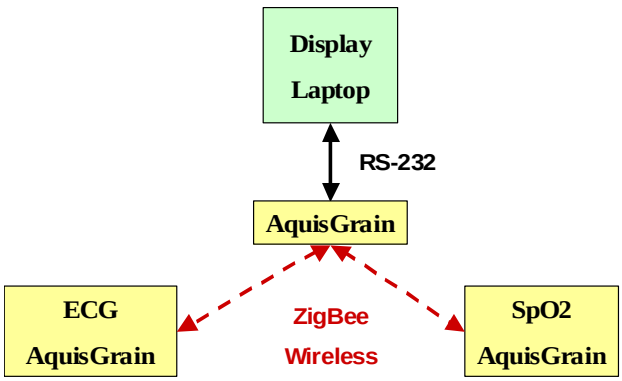


Figure 30. A possible configuration of the demonstrator

5.4 Feasibility and User Satisfaction

The goal of our demonstrator is to intuitively show the feasibility and benefits of using DPKPS-enabled security in the hospital. We also use it to evaluate the performance that the human user perceives on using the system. The evaluated parameters include simplicity, timeliness, transparency, and efficacy.

In the following, we briefly describe the demonstrated scenarios:

- *Security configuration*. The goal of this demonstration is showing the process of configuring DPKPS keying material in a hospital context. The demonstration also shows the simplicity of configuring DPKPS security for a security administrator (no mathematical background is needed).

  The demonstrator system includes the desktop and the two ECG AquisGrain nodes. The ECG AquisGrain nodes include a security software module.

  During the demonstration we show a BioEng generating DPKPS keying material with the Keying Material Generator. The desktop loads the corresponding keying material vector in each ECG AquisGrain using an RS-232 serial port interface.

- *Secure sensor data communication*. The goal of this demonstration is showing the motivation for security in a hospital context. It also shows the property of user-transparent (or unobtrusive) and effective security.

  The demonstrator system consists of a laptop in the role of a clinician PDA and the ECG and SpO2 AquisGrain nodes. The clinician PDA and the ECG belong to the hospital and, then, have been initialized by a BioEng with DPKPS keying material. The ECG AquisGrain simulates a legitimate sensor attached to the patient body. The SpO2 simulates an intruder trying to send bogus SpO2 measurements. The SpO2 does not carry valid keying material. Figure 31 depicts the simulated medical scenario.

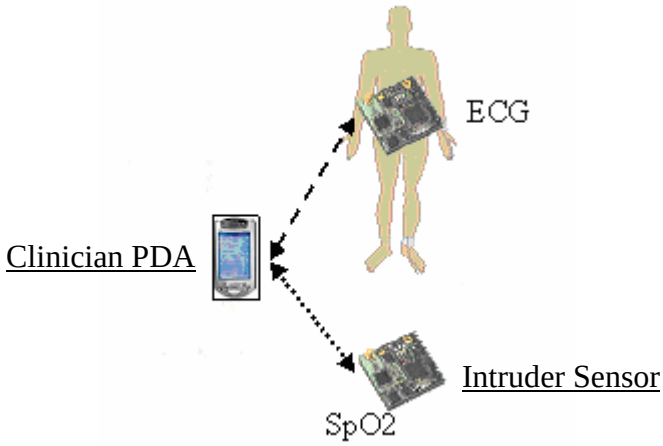


Figure 31. Intruder trying to impersonate a sensor node attached to the patient

We run two security modes: *security off* and *security on*. In the *security off* mode no security mechanisms are activated and, thus, the demonstrator shows the intruder succeeding in sending a false SpO2 measurement of 20%. False SpO2 data together with legitimate ECG data is associated to the patient and displayed on the clinician PDA. Consequently, because a SpO2 of 20% is a value showing bad health status, the clinician may conclude a wrong diagnosis for the patient.

In the *security on* mode, since the intruder SpO2 AquisGrain is not provided of legitimate keying material, it cannot successfully pass the authentication check and, consequently, the false SpO2 measurements are rejected and not displayed by the clinician PDA. The legitimate ECG AquisGrain node succeeds the authentication checking and, therefore, is allowed to send ECG data. The ECG signal of the patient is then displayed on the clinician PDA.

- *Clinician rounds*. The goal of this demonstration is showing security interoperability and flexibility of sensors, PBMs and PDAs belonging to the same security domain. It also shows the property of time efficient security.

  In this scenario we use two laptops and two ECG AquisGrain nodes. The first laptop takes the role of a legitimate PBM and the second, of a legitimate clinician PDA. Both ECG AguisGrain nodes simulate legitimate ECG sensors. The first AquisGrain is attached to patient Heribert and, the second, to patient Oscar.

  We run this demonstrator in the *security on* mode. Clinician David, carrying his PDA, visits patients Oscar and Heribert. Clinician Karin, carrying a PBM, visits the same patients. The demonstrator shows the clinicians monitoring the patient vital signs in the different combinations BSN configurations. That is, (considering a BSN per patient) PDA↔ECG1 and PBM↔ ECG1; PBM↔ECG2 and PDA↔ ECG2; PBM, PDA↔ECG1; and PBM, PDA↔ ECG2.

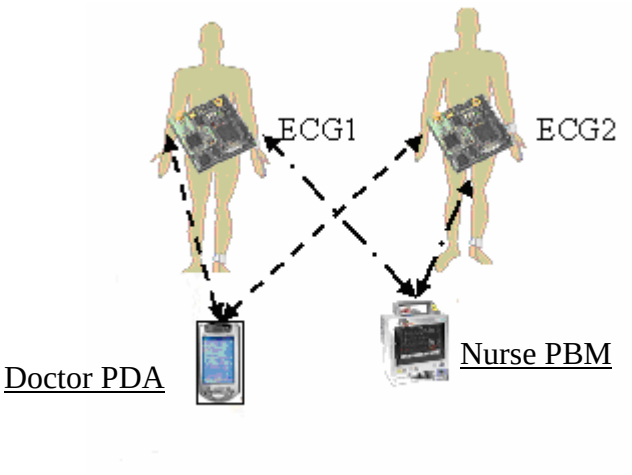


Figure 32. Scenario involving multiple patients and clinicians

  Successful display of vital signs means successful generation of pairwise keys and authentication. The secure communications are established without apparent delays for the clinician.

The following scenarios, to further demonstrate more features enabled by the DPKPS, could be easily shown by slightly extending the current demonstrator:

- *Fine-grained Clinician Access Control*. The goal of this demonstration is showing clinician personalized access control.

During the patient set-up medical procedures a set of sensors are initially configured and attached to the body of a sensor. Included in this set-up, a group of authorized clinician identities can be configured on sensors. Clinician identities shall be associated to DPKPS node IDs.

Now imagine the patient is wearing the legitimate sensors and a clinician approaches with a legitimate PDA. Only in the case that the clinician belongs to the group of users allowed to treat that patient, he will be able to display the vital signs of that patient.

- *Patient @home*. The goal of this demonstration is to show the scalability and flexibility of the DPKPS and also potential usage scenarios beyond the hospital.

  A patient has been treated at the hospital. For instance, he has recovered from a recent heart attack. He is ready to go home but still some of his vital signs (e. g., ECG and SpO2) need to be monitored.

  The hospital provides the patient of a *care box* containing a set of sensors (ECG and SpO2sensors in our example) and a patient storage/relay device (e.g. a UMTS-enabled mobile phone). The patient storage/relay device can be used to regularly store and/or to online relay vital signs. The sensors and the patient storage/relay device have been previously configured within the hospital security domain, i.e. with DPKPS keying material.

  Once at home, the patient (or helped by an assistant) can simply attach the sensors to his body to automatically (plug & play) set-up a BSN. Subsequently, vital signs of the patient are monitored while the patient lives his quotidian life.

  The BSN is protected with security mechanisms, i.e. sensors and devices of the care box establish encrypted and authenticated communications. Then, no intruder can violate patient privacy or safety.

  Occasionally, a clinician comes from the hospital to visit the patient. He carries any PDA, which has been also initialized in the hospital security domain. By using the PDA, the clinician can also *securely* connect to the BSN and monitor the current vital signs of the patient. He may also check the patient vital signs history of the previous days.

  Additionally, the clinician may bring new sensors from the hospital and attach them to the patient's body. This new sensor can establish secure connections with old sensors at the BSN, with the patient storage/relay device and with clinician PDAs.

  The same scenario can be extended to various patients being monitored at home.

5.5 Summary and Conclusions

In this chapter, we have presented the application of DPKPS to provide security in medical BSNs of resource-restricted nodes.

In the medical context, the DPKPS enables the pre-configuration of trust relationships among potential medical wireless devices before they are actually used at the BSN. Further, this setting enables a clinician to attach or to associate two or more such devices and automatically create a patient BSN, without the need to actively configure security for that BSN.

Because DPKPS-based security is plug&play and user transparent, the clinician can exclusively focus on diagnosing and caring the patient.

6 Conclusions and Future Work

A WAHN is a collection of autonomous nodes that communicate with each other by forming a single-hop or, often multi-hop, wireless network and maintaining connectivity in a decentralized manner. The network topology is in general dynamic, because the connectivity among the nodes may vary with itinerating, quitting and joining nodes.

To protect WAHNs against malicious attackers, WAHNs must be provided of authentication, confidentiality and integrity services. Further security services, including user access control and non-repudiation services, may be required by particular WAHN applications.

Performance-aware key management must fundament security services in WAHNs. The WAHN key management service must enable any pair of nodes to establish a key with independence of dedicated security servers, adapting to WAHN dynamics and using cryptographic primitives and protocols adequate to WAHN node resource restrictions.

The HKMI presented in this thesis enables energy-efficient direct key establishment for subsequent use in authentication or confidentiality and integrity services. The HKMI also supports non-repudiation services based on public key cryptography. A PKI underlying the HKMI is used to set-up initial trust of nodes in the WAHN and, thus, generate a random trust graph connecting all the nodes of the WAHN. Nodes of the shortest trust path connecting two end nodes cooperate to securely distribute trust information and symmetric keys to the end nodes.

The HKMI can be applied to support efficient security in applications using relatively dynamic WAHNs of moderate-resource nodes. For instance, the HKMI can be used to protect a WAHN composed of various medical portable wireless actuators around a patient. In such scenario, a potential intruder fails HKMI-enabled authentication verification and, thus, cannot maliciously trigger any actuator. Additionally, because the HKMI supports non-repudiation, the identity of the clinician triggering or setting an actuator is unequivocally registered.

However, because the HKMI is based on public key cryptography, the same solution cannot be applied to WAHNs including extremely resource-constrained nodes. In such a case, key management must be based on lightweight symmetric key cryptography.

The DPKPS presented in this thesis enables lightweight pairwise symmetric key establishment in WAHNs of strictly resource-constrained nodes. In an initial set-up phase a trusted server stores keying material to the nodes. The keying material consists of a number of optimized Blundo's polynomial shares arranged according to a combinatorial distribution. The combinatorial distribution guarantees that any two nodes carry shares of a common polynomial. Blundo's polynomials provide for node authentication.

Afterwards, on deploying the nodes in one (or more) WAHN(s) any pair of nodes initialized with the DPKPS can directly establish a pairwise key without the need for the trusted server to be online.

The DPKPS exhibits excellent scalability, security and energy-efficiency properties. It can be applied to WAHNs of static or mobile nodes and of arbitrary node population size and density. One such application is the protection of medical BSNs for patient vital sign monitoring. In this context, the DPKPS enables plug&play and clinician transparent yet effective security in BSNs. A potential intruder cannot modify any patient vital sign data transferred at the BSN.

Future work includes investigating approaches to increase the resiliency of the DPKPS without relaxing connectivity properties or increasing memory and energy cost. One such approach may explore mechanisms to minimize correlation between keying material stored in nodes associated to the same FPP block.

A key management infrastructure can be designed on the basis of the DPKPS. The key management infrastructure may be further extended with consistent keying material update and revocation mechanisms. Because the ad hoc disconnected nature of MSNs, solving this functionality in this context results a complex problem.

Furthermore, the design of access control mechanisms, including role, group and user access control, in MSNs is also a line of research yet to go. The DPKPS serves the basic identification and keying service on top of which access control services may in the future be defined.

Finally, a WAHN may be formed of devices of diverse resource capability. For instance, a medical WAHN may be composed of a combination of mains-powered devices, portable devices and actuators, and sensors. In this network setting, the HKMI and the DPKPS can be combined within a unified key management architecture. The DPKPS can be used to cover security for sensor-to-any communication and the HKMI to protect communication among more powerful devices. Further work in this line involves the development and testing of such architecture in real WAHN products.

7 References

[1] D. Sanchez and H. Baldus. Hybrid Key Management Infrastructure for Mobile Ad hoc Networks. In Proceedings of the 4th Mediterranean Ad hoc Networking Workshop. 2005.

[2] D. Sanchez and H. Baldus. A Deterministic Pairwise Key Pre-Distribution Scheme for Mobile Sensor Networks. In Proceedings of the 1st IEEE International Conference on Security and Privacy for Emerging Areas in Communication Networks. 2005.

[3] D. Sanchez and H. Baldus. Key Management for Mobile Sensor Networks. In Proceedings of the 1st Mobile Ad hoc Networks and Sensors Workshop. 2005.

[4] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. D. Tygar. SPINS: Security protocols for sensor networks. In Proceedings of MOBICOM, 2001.

[5] A. Pirzada and C. McDonald. Kerberos Assisted Authentication in Mobile Ad hoc Networks. 27th conference on Australasian computer science, vol. 26, pp. 41 – 46. 2004.

[6] S. Capkun, J.-P. Hubaux and L. Buttyan. Mobility helps security in ad hoc networks. In Proceedings MobiHoc'03, 2003.

[7] L. Martucci, C. Schweitzer, Y. Regina Venturini, T. C. Carvalho, W. Ruggiero. "A Trust-Based Security Architecture for Small and Medium-Sized Mobile Ad Hoc Networks". The Third Med-Hoc-Net Workshop, 2004.

[8] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. IEEE Network Magazine, vol. 13, no.6, 1999.

[9] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks, Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000.

[10] H. Luo, P. Zerfos, J. Kong, S. Lu and L. Zhang. Self-Securing Ad Hoc Wireless Networks. 7th International Symposium on Computers and Communications. 2002.

[11] M. Bechler, H.-J. Hof, D. Kraft, F. Pählke and L. Wolf. A Cluster-Based Security Architecture for Ad Hoc Networks. IEEE Infocom 2004.

[12] E. C. H. Ngai, M. R. Lyu. Trust- and Clustering-Based Authentication Services in Mobile Ad Hoc Networks. ICDCSW'04 Workshops - W4: MDC. 2004.

[13] L. Venkatraman and D. P. Agrawal. A Novel Authentication Scheme for Ad hoc Networks. WCNC 2000, pp. 1268-1273, vol.3.

[14] M. Elhdhili, L. B. Azzouz, F. Kamoun. A Totally Distributed Cluster Based Key Management Model for Ad Hoc Networks. The Third Med-Hoc-Net Workshop. 2004.

[15] S. Capkun, L. Buttyan and J.-P. Hubaux. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. IEEE Transactions on Mobile Computing, vol. 2, nº 1, pp. 52-64. 2003.

[16] C. Kaufman, R. Perlman and M. Speciner. Network Security: Private Communication in a Public World. Prentice Hall PTR, 2002.

[17] Network Associates, Inc. An Introduction to Cryptography.

[18] R. Blom. An optimal class of symmetric key generation systems. In Proceedings of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques, pp. 335 - 338, 1985.

[19] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, Perfectly Secure Key Distribution for Dynamic Conferences. In Advances in Cryptology - CRYPTO '92, Springer-Verlag, Berlin, 1993, pp. 471-486.

[20] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In Proceedings of the 9th ACM CCS conference, pp. 41 – 47, 2002.

[21] H. Chan, A. Perrig, and D. Song. Random key pre-distribution schemes for sensor networks. In Proceedings of the 2003 IEEE Symposium on Security and Privacy, p. 197, 2003.

[22] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In Proceedings of the 10th ACM CCS Conference, pp. 42 – 51. 2003.

[23] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In Proceedings of the 10th ACM CCS Conference, pp. 52 – 61. 2003.

[24] J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In Proceedings of the 2nd ACM SASN workshop, pp. 43 – 52. 2004.

[25] S. Zhu, S. Setia and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. ACM CCS Conference, pp. 62-72. 2003.

[26] R. Di Pietro, L. V. Mancini, A. Mei, A. Panconesi. Connectivity Properties of Secure Wireless Sensor Networks. In Proceedings of the 2nd ACM SASN workshop, pp. 53 – 58. 2004.

[27] S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure pebblenets. In Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing, pp. 156-163. 2001.

[28] S. A. Camtepe and Bülent Yener. Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks. In Proceedings of Computer Security- ESORICS, Springer-Verlag, LNCS 3193, 2004, pp 293-308.

[29] J. Lee, and D. R. Stinson. Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In Proceedings 11th International Workshop, SAC 2004, 2004, pp. 294-307.

[30] H. Chan, A. Perrig and D. Song. Key distribution techniques for sensor networks. Wireless sensor networks. Kluwer Academic Publishers, 2004, pp. 277 – 303.

[31] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. Handbook of Applied Cryptography. CRC Press, October 1996.

[32] R. Di Pietro, L. V. Mancini, A. Mei. Efficient and Resilient Key Discovery Based on Pseudo-Random Key Pre-Deployment. In Proceedings of the 4th IEEE Workshop on Algorithms for Wireless, Ad hoc and Sensor Networks. 2004.

[33] D. Dolev and A. C. Yao. On the security of public key protocols. IEEE Transactions on Information Theory. Vol. 29, Issue 2, pp. 198-208. 1983.

[34] J. Newsome, E. Shi, D. Song, A. Perrig. The sybil attack in sensor networks: analysis & defenses. In Proceedings of the third international symposium on Information processing in sensor networks, pp. 259 – 268, 2004.

[35] Network Associates, Inc. An Introduction to Cryptography.

[36] I. Chlamtac, M. Conti and J. J-N Liu. Mobile ad hoc networking: imperatives and challenges. Elsevier Ad hoc Networks Journal. Vol 1 num. 1, pp. 13-64. Elsevier 2003.

[37] Romer, K. Mattern, F. The design space of wireless sensor networks. IEEE Wireless Communications, Vol. 11, Issue 6, pp. 54- 61. 2004.

[38] Fei Hu and Neeraj K. Sharma. Security considerations in ad hoc sensor networks. Elsevier Ad hoc Networks Journal. Vol. 3, pp. 69-89. 2005.

[39] S. A. Camtepe and Bülent Yener. Key Distribution Mechanisms for Wireless Sensor Networks. Rensselaer Polytechnic Institute, Computer Science Department. TR-05-07. 2005.

[40] D. Carman, P. Kruus and B. Matt. Constraints and Approaches for Distributed Sensor Network Security. NAI Labs TR-00-010. 2000.

[41] K. Hoeper and G. Gong. Models of Authentications in Ad Hoc Networks and Their Related Network Properties. University of Waterloo. TR. 2004.

[42] National Institute of Standards and Technology. Wireless Ad hoc Networks Background. Online available at http://w3.antd.nist.gov/wahn_bkgnd.shtml.

[43] M. Jakobsson and S. Wetzel. Security Weaknesses in Bluetooth. In Proceedings of the Conference on Topics in Cryptology: The Cryptographer's Track at RSA, pp: 176-191. 2001.

[44] N. Borisov, I. Goldberg, and D. Wagner. Intercepting mobile communications: The insecurity of 802.11. In Proceedings of MOBICOM. 2001.

[45] T. S. Messerges, J. Cukier, T. A. M. Kevenaar, L. Puhl, R. Struik and E. Callaway. A security design for a general purpose, self-organizing, multihop ad hoc wireless network. In Proceedings of the 1st ACM Workshop on Security of ad hoc and sensor networks, pp. 1-11. 2003.

[46] ZigBee™ Alliance. ZigBee Security Specification Version 1.0. 2004.

[47] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. 7th International Workshop on Security Protocols, Lecture Notes in Computer Science, pp. 172-194.1999.

[48] I. Anderson. Combinatorial Designs: Construction Methods. Ellis Horwood Limited, 1990.

[49] C.J. Colbourn, J.H. Dinitz. The CRC Handbook of Combinatorial Designs. CRC Press, 1996.

[50] W.D. Wallis. Combinatorial Design. Marcel Dekker Inc., 1988.

[51] B. Kaliski. TWIRL and RSA Key Size. RSA Laboratories. Revised May 2003.

[52] M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson and M. Wiener. Minimal key lengths for symmetric ciphers to provide adequate commercial security. A report by an ad hoc group of cryptographers and computer scientists. 1996.

[53] Laywine C.F., Mullen G.L. Discrete mathematics using Latin squares. Wiley, 1998.

[54] T. Falck et al. Advances in Healthcare Technologies. Philips Research Series. Springer Verlag, 2005.

[55] J. Espina et al. Body Sensor Networks. Springer Verlag. To appear in 2006.

[56] IEEE Computer Society. IEEE Standard for Information technology, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)". October 2003

[57] ZigBee™ Alliance. ZigBee Specification Version 1.0. June 2005.

[58] W. Stallings. Cryptography and Network Security: Principles and Practice. Prentice Hall, 2nd Edition, 1999.

[59] ATMEL®. 8-bit AVR© Microcontroller with 128K Bytes In-System Programmable Flash. Atmega128L. 2004.

[60] Chipcon AS *SmartRF*®. CC2420 Preliminary Datasheet (rev 1.2). 2004.

[61] B. Kaliski. TWIRL and RSA Key Size. RSA Laboratories. 2003.

[62] A. Weimerskirch and G. Thonet. A Distributed Light-Weight Authentication Model for Ad hoc Networks. In Proceedings ICISC 2001, pp. 341–354. 2001.

[63] The MSDN Library. http://msdn.microsoft.com/library/default.asp.

[64] The Rijndael Page. http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.

[65] P. Michiardi and R. Molva. Core: A Collaborative Reputation mechanism to enforce node cooperation in Mobile Ad Hoc Networks. Communication and Multimedia Security Conference. 2002.

[66] Eric Holland. Understanding Your Wireless Options. Sensors Magazine, Vol. 21, Nr. 12. Dec. 2004.

[67] J.K. Doyle and J.E. Graver. Mean distance in a graph. Discrete Mathematics Vol. 17, Issue 2, pp. 147-154. 1977.

[68] A. Hodjat, I. Verbauwhede. The Energy Cost of Secrets in Ad-hoc Networks. IEEE CAS Workshop on Wireless Communications and Networking. 2002.

List of Acronyms

AES: *Advanced Encryption Standard*.

BIBD: *Balanced Incomplete Block Design*.

BS: *Base Station*.

BSN: *Body Sensor Network*.

CA: *Certification Authority*.

CPU: *Central Processing Unit*.

DPKPS: *Deterministic Pairwise Key Pre-Distribution Scheme*.

ECC: *Elliptic Curve Cryptography*.

ECG: *Electrocardiogram*.

FPP: *Finite Projective Plane*.

HKMI: *Hybrid Key Management Infrastructure.*

HR: *Heart Rate*.

ID: *Node Identifier*.

KPS: *Key Pre-distribution Scheme.*

MOLS: *Mutually Orthogonal Latin Squares*.

MSN: *Mobile Sensor Network.*

PBM: *Portable Bedside Monitor*.

PDA: *Personal Digital Assistant*.

PGP: *Pretty Good Privacy.*

PKC: *Public Key Cryptography*.

PKI: *Public Key Infrastructure*.

SBIBD: *Symmetric Balanced Incomplete Block Design*.

SKC: *Symmetric Key Cryptography.*

TKD: *Trust and Key Distribution Protocol*.

TL: *Trust Level*.

TP: *Trusted Portal*.

TTP: *Trusted Third Party.*

WAHN: *Wireless Ad Hoc Network*.

WSN: *Wireless Sensor Network*

Detailed Table of Contents