Computer Communications 000 (2016) 1–13

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A privacy-preserving smart parking system using an IoT elliptic curve based security platform

Ioannis Chatzigiannakis[a,b,*], Andrea Vitaletti[a], Apostolos Pyrgelis[c]

[a] Department of Computer, Control, and Management Engineering (DIAG), Sapienza University of Rome, Italy
[b] Computer Technology Institute & Press "Diophantus" (CTI), Greece
[c] Department of Computer Science, University College London (UCL), UK

A R T I C L E I N F O

Article history:
Available online xxx

Keywords:
Smart city
Privacy
Wireless sensor networks
Algorithm engineering
Performance evaluation

A B S T R A C T

Since the initial visions proposed in the SmartDust project fifteen years ago, Wireless Sensor Networks have seen a tremendous development, leading to the realization of the Internet of Things (IoT). Today, there is a large variety of hardware and software to choose from that is easy to set up and use. Even though there is an increasing number of real-world applications that employ large deployments of IoT devices, the wireless nature of communication in combination with the low-end capabilities of the devices raises security and privacy issues that have not been properly addressed. Considering also that sensor node brands are very different in their capabilities, providing a single solution is very challenging.

In this paper we adopt Elliptic Curve Cryptography (ECC) as an attractive alternative to conventional public key cryptography, such as RSA. ECC is an ideal candidate for implementation on constrained devices where the major computational resources, i.e., speed, memory are limited and low-power wireless communication protocols are employed. That is because it attains the same security levels with traditional cryptosystems using smaller parameter sizes. We provide a generic implementation of ECC that runs on different host operating systems, such as Contiki, TinyOS, iSenseOS, ScatterWeb and Arduino. Furthermore, it runs on smartphone platforms such as Android and iPhone and also any linux based systems (e.g., raspberryPi). Our implementation does not contain any platform-specific specializations, allowing a single implementation to run natively on heterogeneous networks.

We look into the Smart Parking application domain and provide a solution that protects the privacy of the users by totally avoiding the exchange of confidential information. We also show how to protect a user's privacy by adapting the tool of zero knowledge proofs (ZKP) with our ECC implementation. We study the performance of our system in an real-world outdoor IoT testbed and analyze the execution time and network overhead for each available hardware platform. Our code is available as open source software and can be used from developers who wish to achieve certain levels of security and privacy in their applications.

1. Introduction

The last decade we witnessed a tremendous progress towards the interconnection of the digital and physical domains, giving rise to the "Internet of Things". ICT is increasingly being embedded into the physical world: smartphones, NFC, RFID, and, networked sensors are now common items in our everyday lives. The exponential growth of connected objects that can participate in the IoT ecosystem is expected to reach 5.8 billion by 2020[1,2] and enables an ever-growing gamut of application domains and innovative services that will change the way we live, work and communicate. It is foreseen that in the near future, everything from individuals, groups, objects, products, data and services will be connected and impacted by the IoT paradigm.

As individuals interact directly with the IoT ecosystem, huge amounts of data are being recorded and then shared, aggregated, annotated, stored, processed and finally consumed. It is clear that collected data may be used to extract or infer sensitive

* Corresponding author. Tel.: +39 0677274073; fax: +39 0677274002.
E-mail addresses: ichatz@dis.uniroma1.it (I. Chatzigiannakis), vitaletti@dis.uniroma1.it (A. Vitaletti), apostolos.pyrgelis.14@ucl.ac.uk (A. Pyrgelis).

1 Ericsson, "More than 50 billion Connected Devices", Report, Feb. 2011.
2 GSM Association, The Mobile Economy 2013 online report: http://www.gsmamobileeconomy.com/GSMA%20Mobile%20Economy%202013.pdf

http://dx.doi.org/10.1016/j.comcom.2016.03.014
0140-3664/© 2016 Elsevier B.V. All rights reserved.

information about users' private lives, habits, activities and relations, which all refer to individuals' privacy [1,2]. It is therefore crucial that IoT systems must guarantee the confidentiality and integrity of the information and the privacy and anonymity of users. IoT enabled systems must respect the context in which personally identifiable information is collected, and ensure that end-users are in control of sensitive data and are able to determine for themselves when, how and to what extent, information about them are communicated to others [3,4].

Challenge 1: Resource-constrained devices. The basis of IoT is the ability to integrate sensing, computation and wireless communication in small, low-power devices that can be seamlessly embedded in complex physical environments. Such low-sized embedded devices have limited sensing, signal processing, and communication capabilities and are usually battery operated. Due to this resource-constrained environment of operation, applying standard security and privacy requirements is extremely challenging. As an example consider that some smart devices have limited computing and storage capabilities, thus cryptographic algorithms and protocols that require intensive computation, communication, or storage are simply not applicable. It is too costly (in terms of computation) to authenticate using a public key and too costly (in terms of memory and computation) to store one-way chains of keys. Also consider that some smart devices may be battery operated, forcing security mechanisms to reduce their energy consumption. These constraints greatly increase the difficulty of securing IoT-enabled systems and make them more vulnerable to security threats. Still, since building secure IoT-enabled systems is of paramount importance, the only viable solution is to combine different techniques for securing the system, i.e., implement secure routing schemes, secure aggregation, provide group key establishment methods, cryptographically encrypt messages etc.; although each single level defense mechanism is highly vulnerable, the combination of multiple attacking angles increases the overall achieved security.

Challenge 2: Heterogeneous networks. The IoT success story has led to a serious practical issue that has not been sufficiently addressed in the past: IoT node brands are very different in their capabilities. Some nodes have 8-bit microprocessors and tiny amounts of RAM, while others burst with power, being able to run desktop operating systems such as Linux. Consequently, the software running on these systems is very different on the various nodes. Even worse, the operating systems on most IoT nodes provide barely enough functionality to implement simple cryptographic algorithms. The developer is forced to spend great attention on low-level details, making the process painfully complex and slow. Still, while it is easy to write code for a specific platform, the inherent heterogeneity of IoT-enabled systems requires the integration of multiple platforms and operating systems. Inevitably the developer is forced to either develop platform-independent code (a very challenging task) or maintain parallel branches of the application code for each different architecture (a very time consuming task). It is therefore crucial to provide tools that help the developer implement the algorithm once and use the code in different platforms across the application ecosystem.

Challenge 3: Absence of common standards. The vision of the IoT has led to substantial standardization progress across different bodies (e.g., IETF CoAP/RPL/6LowPAN/6TSCH, IEEE 802.15.4e, ETSI M2M, 3GPP MTC, oneM2M), providing technical solutions tailored to the resource-constrained embedded nodes, ranging from the lower to the upper OSI layers. Yet, until now, no standard has managed to attract the vast majority of the stakeholders and dominate the domain. Current IoT deployments are, more often than not, privately run to serve a specific application, enforcing a tight association between the application, the network used by the application and the sensors that constitute that network. Clearly developing security and privacy requirements for application-specific and mission-oriented systems where no single protocol stack is used is extremely challenging. Most attempts are inherently non-scalable, exhibit low cost-efficiency, are non-adaptive and usually require tremendous efforts to integrate them with existing and well-established services. Cryptographic primitives and security architectures need to be generic enough in order to be implementable in a variety of systems supporting different communication primitives.

Existing approaches. Up till now the majority of the cryptosystems proposed in the relevant bibliography were studied using theoretical tools. Novel techniques have been proposed, that for example, target small sized keys, reduce communication exchanges, operate under the assumption of insecure communication channels, etc. More often than not, in these theoretical studies, researchers tend to design an algorithm in an abstract way. This happens because an algorithm should be able to be used in many different situations and it is up to the developer to decide the way it should be turned into code for a real system. As a result, almost every time the developer finds many limitations in the ways she can operate within the given hardware and software specifications. These problems are further augmented when implementing algorithms for wireless sensor networks due to the extremely limited resources and also due to the heterogeneous nature (both in terms of hardware and software). Algorithm development for such networks is complex as it unites the challenges of distributed applications and embedded programming [5]. During the past years, a limited number of efforts have been made to present the difficulties addressed while converting theoretical algorithms into code by focusing on a single hardware platform and a single network stack, e.g., [6,7]. Moreover, these experimental efforts focus on the evaluation of the operations of the protocol without focusing on specific application requirements derived from real-world needs. Evaluating the performance of a cryptosystem is certainly a very challenging and important step towards creating tools and code libraries. However, we strongly believe that providing solutions that can be integrated into real-world systems requires a combined approach where the requirements of the application are taken into consideration both during the implementation of the code library as well as during the evaluation process.

The fact that the security is not often built directly into inexpensive sensor devices but considered as an after-thought is emphasized in the very recent report by Atmel on security issues for embedded systems [8]. The report indicates that there is an urgent need to deliver cost effective solutions that enable robust security but also to retain the flexibility to deliver real benefits in the face of expected threats. This requires well-architected and interoperable frameworks across vendors and technologies, integrated at an IP and silicon level to enable the evolution of security services the whole industry can leverage. Our work is motivated by this need for practical tools that move away from RSA to elliptic curves based cryptosystems.

Our approach. In contrast to existing experimental approaches, we make a step towards providing a generic implementation (written in C++) that runs on a number of IoT platforms. Our design effort is driven by the requirements and specification of real-world city-scale infrastructures and smart city applications. We implement our security scheme and the cryptographic protocols in a way such that they can be compiled in a very broad variety of IoT systems without changing any line of code. Our goal is to provide a privacy-preserving IoT application environment that addresses the following issues:

Platform independence. Code can be compiled on a number of different hardware platforms, usually without platform-dependent configurations, i.e., no "#ifdef" constructions.

OS independence. Code can be compiled for different operating systems. This includes systems based on C like Contiki, C++ (the iSense firmware), nesC (TinyOS) and also more feature-rich environments (the Android and iOS) and also standard systems (linux based systems).

Exchangeability. Components can be exchanged with other implementations without affecting the remaining code. Moreover, both generic components and highly optimized platform-specific components can be used simultaneously.

Cross-layer algorithms. An algorithm can be composed out of other algorithm, thus enabling the use of existing algorithms for the implementation of more complex ones. Moreover, we can stack protocols on top of each other, extending their functionality.

Scalability and efficiency. Code is capable of running on a great variety of hardware platforms, with CPUs ranging from 8-bit microcontrollers to 32-bit RISC CPUs, and with memory ranging from a few kilobytes to several megabytes. Algorithms need to be very resource-friendly on the platforms from the lower end, and at the same time be able to use more resources if available.

To our knowledge, our solution is the only successful attempt to achieve all of these goals at once. Our system runs on IoT systems comprised of heterogeneous hardware platforms that incorporate different operating systems.

In order to demonstrate the generality and reusability of our solution, we focus on the Smart City domain that has attracted a lot of attention mainly due to rapid growth of cities making them the main driver of global environmental changes: cities, occupy only 2% of the earth landmass, consume about 75% of the world's energy and produce 80% of its greenhouse gas emissions [9]. According to the United Nations[3], the urban populations will grow by an estimated 2.3 billion over the next 40 years, while as much as 70% of the world's population will live in cities by 2050. Such a dramatic expansion of the cities has brought to focus the need to develop cities in a sustainable manner, while also making the quality of life in the cities better. Due to these factors, a wide range of problems have been tackled by exploiting IoT ecosystems and their use in the Smart City concept has matured significantly. However we have to keep in mind that as IoT infrastructures pervade our everyday life, it is imperative that our privacy is protected.

In this work we focus further on the particular case of Smart Parking management systems. The optimization of parking spaces within a city has become a necessity given the limited space available and the fact that searching for a parking spot may generate up to 40% of the total traffic in a city district[4]. A large number of smart parking systems have been proposed in the literature (e.g., some of them focusing on the guidance of the citizen to a free parking slot [10], others focusing on the automation of the operation and management [11]) and many systems have left the research labs and are now available as commercial systems. Interestingly, Smart Parking management systems raise many crucial issues related to the privacy of citizens and the confidentiality of their data. Consider that existing Smart Parking systems heavily rely on the citizens giving away their location information. Location information is communicated across various parts of the infrastructure and it is eventually stored at a central database. Also consider that in many cities and communities, zoning code parking regulations dictate that certain parking spaces are shared among residents of specific neighborhoods or are reserved for citizens with special needs. Clearly such confidential information needs to be secured: it could be embarrassing to be seen at certain places, with the natural assumptions that follow from proximity to an abortion clinic, crack house, AIDS clinic, business competitor, or political headquarters [12].

A second interesting dimension of Smart Parking systems is their dependence on IoT infrastructures that monitor the status of the parking spaces. During the past couple of years a few cities decided to install IoT devices, such as Santander where about 400 IoT devices where installed under the asphalt communicating over 802.15.4[5], while others use crowdsourcing technics that rely on the citizens smartphones, such an example is Amsterdam[6]. In these deployments the resulting system is composed of federated networks comprised of hardware devices of a wide range of capabilities. These installations demonstrate that establishing an end-to-end security framework is extremely challenging and in many cases requires implementing (and maintaining) of cryptographic mechanisms for multiple hardware platforms.

Based on these observation, in this paper we present a new approach that totally avoids the need to exchange private information. In our system citizens do not let location information (or other confidential information) leave their local device. We implement our solution by carefully applying a platform-agnostic approach thus having a single implementation for all the security mechanisms employed throughout the different network stacks and hardware platforms. In this way our solution succeeds in both goals: (a) it protects the citizens' privacy and totally avoids storing confidential information; (b) it keeps code maintenance costs at minimum levels – any improvements to the cryptographic mechanisms are done once for all platforms and networks. In order to make our solution applicable to both embedded systems and cloud servers we adopt the elliptic curve version of Diffie-Hellman problem [13] – a generic implementation that offers the same level of security as other public key cryptosystems, using smaller key sizes. Our implementation allows us to use much smaller keys than conventional, discrete logarithm based cryptosystems (an 160-bit key in an elliptic curve cryptosystem provides equivalent security with a 1024-bit key in a conventional cryptosystem as proposed by NIST). A reader can realize that using elliptic curve groups instead of multiplicative groups, the arithmetic operations need less time to execute, less memory space and thus less energy consumption. We believe that due to this fact, elliptic curves are the only reasonable choice for resource-constrained IoT networks, where the resources are very limited. In fact, it has been proven that elliptic curve cryptosystem (ECC) actually outperforms RSA on constrained environments in terms of computation time, memory requirements and thus energy consumption [14]. Moreover, ECC offers the advantage of smaller message sizes that cost less and have better chances of being delivered.

We look into a real-world deployment of IoT devices in a Smart Parking environment and demonstrate how it successfully protects the citizens' privacy by adapting the tool of zero knowledge proofs (ZKP) based on the Eliptic-Curve cryptography implementation. We conduct a thorough evaluation of our system for different low-end microcontrollers (e.g., Jennic JN5139, IT MSP430). We provide real-world evidence in terms of execution time, message exchanges and memory requirements of our platform. We highlight the advantages and point out the disadvantages of our approach and provide some valuable technical insights related to IoT hardware and

[3] United Nations, "World Urbanization Prospects", 2009.

[4] No Vacancy: Park Slope's Parking Problem X – http://www.transalt.org/news/releases/126

[5] Libelium WaspMotes and Meshlium products installed at Santander – http://www.libelium.com/smart_santander_parking_smart_city/

[6] Mobipark solution – http://amsterdamsmartcity.com/projects/detail/id/64/slug/smart-parking

networking technologies. Our code is available as open source software and can be used from developers who wish to achieve certain levels of security and privacy in their IoT systems.

2. Smart parking services

As cities get more and more *intelligent*, the concept of smart parking is becoming reality. Around the world, thousands of businesses and public authorities that offer car-parking facilities are constantly striving to improve quality, convenience and choice. Smart Parking services provide new ways to optimize parking space usage, improve the efficiency of parking operations and help traffic in the city flow more freely. A good variety of web-based parking management solutions are available in the market, that establish advanced parking services by utilizing smartphones technologies for end-user interaction, and relying on ultra low power wireless mesh networks deployed across the city. Typical examples of such services include: provisioning of real time parking occupancy status, automatic enforcement of zoning code parking regulations, handling of fees as per the zone charges, etc. A critical aspect of existing systems is that while interacting with them, citizens are required to reveal a certain number of private information (e.g., address of residency, working address, etc.). In addition, as history of the transactions is stored in the cloud for long periods of time, patterns of behavior can be extracted by simply examining the stored data. It is evident that the provision of advanced parking services raise important privacy issues as untrusted entities could extract from the network private information about the citizens (e.g., Mr. Smith has just arrived home) and potentially derive patterns of behavior without their consent (e.g., to better target marketing content).

A Smart Parking solution should provide all the advantages discussed above without disclosing private information on the user and/or his/her location. The need for privacy-preserving location and the risks if location data leaks to an unscrupulous actor are discussed in [12].

In contrast to existing solutions, we follow a totally different approach. We envision solutions that totally avoid storing confidentially and history data on the cloud and rely only on local interactions to guarantee the proper operation. Our solution completely avoids disclosing any sensitive information on the users (e.g., on their location) thus these information cannot be compromised by the system. Any attempt to extract confidential will have very limited success, and will also require malicious users to employ physical surveillance techniques at multiple points of the infrastructure (and during specific times) to extract such information thus significantly increasing the cost and potential success of an attack.

In this work we abstract smart parking management applications as solutions that are privacy-preserving in the sense that (i) sensors embedded in the parking places, (ii) mobile applications running on smartphones and (iii) city authorities/inspectors interact in a way such that:

- Users can provide proof of payment for the parking spot that they use to inspectors without revealing critical information. Different flavors of parking strategies are supported: time-based, vehicle-size based etc.
- The system is compatible with Zoning code parking regulations. Users that are entitled in using/occupying a parking zone/spot (e.g. a reserved parking for people with disabilities) can easily provide a privacy-preserving proof of their ability to use the parking space.
- The parking system can be integrated with the transportation services. Users can claim discounts for using the public transportation on days that they decided to leave their car. This can be used to incentivize the use of intermodal car parks and platform exchanges.

Note that since our system follows a platform-agnostic implementation approach, our privacy-preserving design can be extended to cover other parking management services.

3. Related work

As IoT infrastructures are deployed and smart city services become integral parts of our lives, a number of issues related to privacy and trust need to be addressed. Data confidentiality and authentication, access control within the IoT network, privacy and trust among users and things, and the enforcement of security and privacy policies are among these issues. However, the different standards and communication stacks involved in combination with the wide variety of embedded hardware components make traditional security countermeasures difficult to be directly applied in the IoT domain. Moreover, the high number of interconnected devices arises scalability issues; therefore a flexible infrastructure is needed to be able to deal with security threats in such a dynamic environment.

For the smart parking application domain, a good variety of solutions have been deployed in pilot studies and evaluated in experimental infrastructures. Unfortunately (to the best of our knowledge) no existing solution properly address the privacy issue related to such a smart city service. We here include some examples of systems that employ different technologies (e.g., RFID, SMS, etc.) to implement smart parking services.

In [15] the authors present a reservation-based smart parking system. In this solution, the mobile phone is used to provide information on the driver's identity to the reservation authority of the parking lot without any formal grantee of privacy. A smart parking reservation system employing short message services (SMS) is presented in [16]. Also in this case, the driver has to send an SMS providing both information on her identity (i.e. the mobile phone number) and on the location of the parking she wants to occupy. In some solutions, an RFID tag is associated with the vehicle and is automatically detected and identified by RFID readers installed at the entry and exit points [11]. In [17] the authors present a similar system based on image recognition. In both [11] and [17] different technologies are used to identify a driver and then to allow/deny access to a smart parking, namely both the identity and the location of a user are disclosed.

One could argue that given the above systems, we could improve the security levels and protect the privacy of the users by employing popular identity protocols and crypto-frameworks. The applicability and limitations of existing IP-based Internet security protocols and other security protocols, which are potentially suitable in the context of IoT infrastructures are examined in [18]. The analysis indicates that public key cryptography is more suitable for IoT systems (over symmetric approaches), provided that the associated asymmetric techniques are properly optimized. Furthermore, asymmetric techniques are more suitable for the local and distributed solution we propose, because they more effectively address the issue of key distribution in this setting. Heavyweight cryptographic operations, i.e., based on RSA and Diffie-Hellman agreement protocols should be replaced by light-weight operations, i.e., using symmetric cryptography or applying more lightweight asymmetric primitives such as ECC and NTRU. Indeed security cryptography techniques and key management (like Elliptic curve cryptography-ECC and Pairing-based Cryptography-PBC) are computationally feasible in resource-constrained devices in the IoT allowing on top of them more complex security methods as Identity-based encryption (IBE) [19] and Attribute-based encryption (ABE) [20].

In the past a number of implementations of public key cryptography based on elliptic curves was developed for low-power wireless networks [14,21,22]. These implementations provide further evidence that ECC is a suitable approach for establishing secure IoT-based systems. We note that these implementations are platform specific (e.g., for TinyOS platform) and also are integrated within the particular network management algorithms (e.g., for secure routing) thus making the particular implementations hard to apply to another platform or another part of the network stack.

Another approach would be to use one of the privacy schemes that have been proposed for wireless sensor networks after suitably applying it to IoT hardware devices. However, it appears that a unique and well-defined solution able to guarantee confidentiality in a IoT context is still missing, as also asserted in [23]. It is worth to note that many efforts have been conducted in the WSN field [24–27], but several questions arise: existing proposals do not address the heterogeneous nature of IoT environments and the need to support different application contexts; mechanisms are not reusable; they do not ensure an end-to-end integrity verification mechanism in order to make the system more resilient to malicious attacks. Very recently, a thorough examination of privacy and trust mechanisms was presented in [28]. The analysis concludes that existing solutions currently lack an unified vision, able to respond to all the IoT requirements, both in terms of security and privacy and network performance. They point out that the need for interoperable solutions that rely on independent distributed components, able to interact and cooperate with each other and also to exchange data on the basis of standards.

Our work is motivated by the Standard Template Library (STL), the Computational Geometry Algorithms Library (STL) [29], and Boost [30] that have a long-standing tradition on desktops and servers. They share a great programming concept that we heavily use for our platform: using C++ templates, one can construct complex object-oriented software architectures that can be parameterized for many different applications. The price of generality is paid at compile time. The final binary contains highly efficient and specialized code, so that there is no overhead at runtime.

The situation for IoT systems is not as promising. There have been approaches to overcome the issues of incompatible nodes by providing generic operating systems that run on multiple platforms. Examples are Contiki [31] and TinyOS [32]. Neither runs on all platforms we are envisioning. Even worse, both introduce new programming paradigms that are valid only for the specific targets, such as protothreads in Contiki, and the whole programming language nesC [33] of TinyOS. The C-inspired nesC attempts to allow for the construction of component architectures with early binding (that share common points with our approach), but achieves this through introducing a new language that requires a custom compiler.

Another challenging issue is the heterogeneity of hardware (see discussion above). It is very simple to have nodes exchange messages if they are of the same kind, and with the same operating systems. It becomes surprisingly hard to let nodes of different brands communicate with each other, even if both of them use standardized IEEE 802.15.4 radios. A promising approach is the Rime Stack [34,35], a layered communication stack for sensor networks. It runs only on Contiki. Sauter et al. [36] demonstrated that is possible to communicate between sensor nodes running Contiki and TinyOS. Since TinyOS uses IEEE 802.15.4, the Rime Stack and Chameleon Module had been modified on Contiki. More recently, the mkSense library presented in [37] enables IoT application development over heterogeneous IEEE 802.15.4 networks (including ATmega328, Jennic JN5139, IT MSP430 and ARM 9 processors).

Another attempt to produce a well-defined environment that runs on different platforms was proposed by Boulis et al. [38]: SensorWare defines a custom scripting language; its syntax is based on Tcl. Consequently it focuses on richer platforms with at least 1 Mbyte of ROM and 128 KBytes of RAM. A similar approach is Maté [39], a virtual machine running on top of TinyOS. It targets also small devices with a very limited amount of resources, using a custom assembler-like language.

Not surprisingly, there are also attempts to run a Java Virtual Machine (JVM) on sensor nodes [40]. Squawk [41] is a JVM by Sun Microsystems that runs on Sun Spots. Obviously such an approach is not suited for low-end sensor nodes, and also not for time-critical applications.

4. Privacy preserving outdoor parking management

We proceed by presenting our smart-parking application that allows citizens to park without the need to reveal sensitive personal data (e.g., the address of residence, if they are people with disabilities etc). We completely avoid using power-consuming encryption technics to protect the data that are transmitted over the IoT infrastructure between the parking sensors and the cars. We also avoid using wireless encryption techniques that provide low levels of protection. Our application completely avoids transmitting over wireless and/or untrusted networks services information that could reveal the location of the citizens, the address of their residence or whether they belong to a specific citizens groups. We also completely avoid storing sensitive private information at cloud-based storage spaces.

In contrast to existing solutions, we incorporate privacy at the application design level, and thus information stored cannot be exploited to reveal the citizens' location or detect patterns of behavior. We follow the ABC4Trust[7] reference architecture and design an application that allows different Security crypto-mechanisms to coexist, be interchanged and federated. We implement our design using the WiseLib[8], a generic and platform independent algorithmic library [42] that allows different IoT technologies to form heterogeneous infrastructures. Our application uses a *Zero-Knowledge* protocol to prove the possession of a piece of information without revealing it. Since no personal information is communicated in the network – but just a proof that this information is valid – the confidentiality of the data is easier to guarantee.

We assume that the city maintains a secure database where private information of citizens are stored (e.g., their residence, if they belong to a special group etc). This database does not need to be connected to the Internet or to the Smart Parking system. We assume that this database is operated and maintained by an authority which we call the *"Issuer"*[9]. We assume that the citizens trust the *Issuer* for protecting the confidentiality of their data.

Use Case 1: Citizen enrolled in the system. Initially, the citizens (i.e., the *User* of the system[10]) are required to contact the *Issuer* in order to acquire a unique private credential (i.e., a private key) that will be used by the smart-parking application. The *Issuer* authenticates the *User* using an external mechanism (either online, or offline requiring the *User* to physically present their identity at the *Issuer*'s premises) and generates the credential of the *User*. The credential encodes the *User*'s attributes (name, vehicle registration plate and some fresh random value) and their correctness is guaranteed by the *Issuer*. The human *User* is represented by her *User Agent*, a software component running either on a local device (e.g., on the

[7] FP7-ICT-257782. Project Attribute-based Credentials for Trust (ABC4Trust). http://abc4trust.eu

[8] FP7-ICT-224460. Project Wisebed. http://wisebed.eu

[9] We here follow the ABC4Trust terminology. In the identity management literature, the *Issuer* is also referred to as the identity provider or attribute authority.

[10] We here follow the ABC4Trust terminology. In the identity management literature, the *User* is also referred to as the requester or the subject.

User's smart phone, or on a special hardware NFC/RFID token to which credentials can be bound to improve security).

Use Case 2: Parking in a restricted-access zone. The Parking sensors (which we call the *Verifier*[11]) are responsible for protecting access to the parking spaces by imposing restrictions on the credentials that the *Users* must own and the information from these credentials that the *Users* must present in order to be allowed to use the parking spot. When a *User* is approaching (the car is detected via the ferromagnetic technology) the *User* generates from her credentials a presentation token that contains the required information and the supporting cryptographic evidence. This is transmitted to the *Verifier* via the NFC/RFID wireless medium. Essentially the *User* needs to prove two things: first that she is a valid citizen and second either that she is entitled to park in this spot (e.g., lives on neighborhood, without revealing her address) without charge or that she has payed.

Use Case 3: Integrated services. The *User* wants to claim a discount for using the public transportation since she decided to leave the car in the parking zone. Once the *User* has been authenticated to use the parking zone, the *user* agent generates a second public key that encodes her right to claim a discount based on the credentials available. The new public key is transmitted to the *Verifier* that digitally signs it and returns it to the *User*. From now on the *User* can user the public transportation services and present the newly generated certificates to the *Verifier* agent (possibly running on the ticket validation machine on the bus/train/tram/metro). This verification takes place by showing that two digitally signed by the corresponding authorities public keys have the same discrete logarithm. If the verification process succeeds the citizen gets a discount on her ticket. A malicious user, is not able to extract information about citizen's private data (e.g. her name) or to get product discount if she does not match the necessary criteria.

5. Zero knowledge protocols for smart city IoT infrastructures

Generally, a zero-knowledge protocol allows a proof of the truth of an assertion, while conveying no information whatsoever about the assertion itself other than its actual truth [43]. Usually, such a protocol involves two entities, a prover and a verifier. A zero-knowledge proof allows the prover to demonstrate knowledge of a secret while revealing no information whatsoever of use to the verifier in conveying this demonstration of knowledge to others.

In order to better understand which zero-knowledge protocol is more suitable for our smart parking application, we consider in this work instances of interactive proof systems and non-interactive proof systems. In the first category, a prover and a verifier exchange multiple messages (challenges and responses), typically dependent on random numbers which they may keep secret whereas in the second the prover sends only one message. In both systems the prover's objective is to convince the verifier about the truth of an assertion, e.g. the claimed knowledge of a secret. The verifier either accepts or rejects the proof. We implement these protocols and conduct a thorough evaluation of their performance in a real-world IoT deployment. Our goal is to identify the most suitable for IoT-enabled smart city applications and in particular to our Smart Parking case study.

A zero-knowledge proof must obey the properties of completeness and soundness. A proof is **complete**, if given an honest prover and an honest verifier, the protocol succeeds with overwhelming probability and **sound** if the probability of a dishonest prover to complete the proof successfully is negligible [44]. Additionally, a protocol which consists a proof of knowledge must have the **zero-knowledge property**: there exists an expected polynomial-time algorithm which can produce, upon input of the assertions to be proven – but without interacting with the real prover, transcripts indistinguishable from those resulting from interaction with the real prover.

A wide variety of zero-knowledge protocols based on the Discrete Logarithm Problem (DLP) has been proposed so far, e.g. in [45,46]. The Discrete Logarithm Problem is defined over arbitrary cyclic groups. A common example of cyclic group is the multiplicative group Z_n^* of order n, where n is a prime number and the group operation is multiplication modulo n. In such a group the Discrete Logarithm Problem (DLP) can be defined as follows: Given a prime n, a generator g of Z_n^* and an element $b \in Z_n^*$, find the integer x, $0 \le x \le n-2$ such that $g^x = b (mod\, n)$ [43].

Another common example of cyclic groups are elliptic curve groups which are defined over an additive group F of order n (note that n is no longer necessarily a prime number). The analogous problem to DLP over elliptic curve groups is called ECDLP (Elliptic Curve Discrete Logarithm Problem) and can be defined as follows: Given an elliptic curve E over a field F of order n (referred to as F_n from now on), a generator point $G \in E/F_n$ and a point $B \in E/F_n$ it is computationally hard to find x such that $B = x \cdot G$.

We now proceed by presenting the adaptation of five zero-knowledge protocols based on the DLP using the Elliptic Curve Discrete Logarithm Problem (ECDLP). This adaptation is a key step for porting such protocols to Smart City IoT infrastructures due to the Elliptic Curve Cryptography (ECC) advantages. It is well established that ECC can offer the same level of security as other public key cryptosystems, using smaller key sizes. This fact makes it suitable for implementations that concern constrained environments as it saves computational time and memory space and consequently reduces energy requirements. Such restrictions consist the real challenges when considering implementations on embedded devices.

5.1. Zero knowledge proof of discrete logarithm with coin flip

One of the first zero-knowledge protocols of discrete logarithm that was originally presented in [47]. Its elliptic curve analogous is as follows: Given an elliptic curve E over a field F_n, a generator point $G \in E/F_n$ and $B \in E/F_n$ Prover wants to prove that he knows x such that $B = x \cdot G$, without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover sends the point A to Verifier
- Verifier flips a coin and informs the Prover about the outcome
- In case of HEADS Prover sends r to Verifier who checks that $r \cdot G = A$
- In case of TAILS Prover sends $m = x + r (mod\, n)$ to Verifier who checks that $m \cdot G = (x + r) \cdot G = x \cdot G + r \cdot G = A + B$

The above steps are repeated until Verifier is convinced that Prover knows x with probability $1 - 2^{-k}$ for k iterations.

Why it works: The protocol works as expected because in each iteration the steps to be executed depend on the outcome of the coin that the Verifier flips and the Prover cannot affect this. It needs to be executed for many iterations in order for the Prover's cheating probability to become very small. A dishonest Prover in each iteration can be prepared for only one of the coin outcomes and thus his cheating probability is 1/2. For example, if he prepares for TAILS he can generate a random m, compute $A = m \cdot G - B$ and send this point A to Verifier. But if HEADS come up this attack will not work. That is because he will need to compute a value $r \in F_n$

[11] We here follow the ABC4Trust terminology. In the identity management literature, the Verifier is also referred to as the relying party, the server, or the service provider.

that generates A and that is an instance of the ECDLP. Thus, after k iterations, the Verifier is convinced with high probability $(1-2^{-k})$ that the Prover is honest.

5.2. Schnorr's protocol

An improvement of the previous protocol was originally presented in [45]. The elliptic curve version of Schnorr's protocol, slightly modified, is the following: Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$. They both know $B \in E/F_n$ and Prover claims he knows x such that $B = x \cdot G$. He wants to prove this fact to Verifier without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover sends the point A to Verifier
- Verifier computes random $c = HASH(G, B, A)$ and sends c to Prover
- Prover computes $m = r + c \cdot x (mod\, n)$ and sends m to Verifier
- Verifier checks that $P = m \cdot G - c \cdot B = (r + c \cdot x) \cdot G - c \cdot B = r \cdot G + c \cdot x \cdot G - c \cdot x \cdot G = r \cdot G = A$

Why it works: This protocol is superior to the previous one as it needs to be executed for one round. Verifier's coin flips (in correspondence with the Coin Flip protocol) are simulated using a hash function known only to him. A dishonest Prover has a tiny chance of cheating as he would have to fix the value of $P = m \cdot G - c \cdot B$ before receiving Verifier's hash value c. Under the assumption that the hash function used by the Verifier is secure, a Prover who does not know x, the discrete logarithm of B, cannot cheat.

5.3. Transforming Schnorr's protocol to digital signature

In [48], the authors propose that with the use of a hash function and an agreement on an initial message m one can remove the interactivity from such protocols. The Verifier's random choices can be replaced with bits produced by a secure hash function. Thus, the next protocol is proposed.

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$, a point $P \in E/F_n$ that represents the message the Prover wants to send and a hash function HASH (e.g. SHA-1). They both know $B \in E/F_n$. The Prover claims that he knows x such that $B = x \cdot G$ and he wishes to prove this fact to Verifier without revealing x.

Protocol Steps:

- Prover generates random $r \in F_n$ and computes the point $A = r \cdot G$
- Prover computes $c = HASH(x \cdot P, r \cdot P, r \cdot G)$
- Prover computes $s = r + c \cdot x (mod\, n)$
- Prover sends to Verifier the message: "$s||x \cdot P||r \cdot P||r \cdot G$"
- Verifier computes $c = HASH(x \cdot P, r \cdot P, r \cdot G)$
- Verifier checks that $s \cdot G = (r + c \cdot x) \cdot G = r \cdot G + c \cdot x \cdot G = r \cdot G + c \cdot B = A + c \cdot B$
- Verifier checks that $s \cdot P = (r + c \cdot x) \cdot P = r \cdot P + c \cdot xP$

Why it works: In this protocol we apply the non interactiveness trick proposed in [48]. The Prover simulates both the Prover and the Verifier with the use of a hash function and publishes the transcript of this whole dialogue. This way the Prover sends only one message and the Verifier either accepts or rejects. The Prover generates a random number as in previous protocols but the Verifier's random choices are simulated by hashing the input along with a value calculated from the Prover's choice of r. Thus, the Verifier's random choice depends on Prover's random choice and it is made hard to fake the outcome. The value c is really a challenge for the Prover as it is computed from the hash function and it is out of his control. If the Prover does not know x, in order to cheat he would try to find s satisfying $s \cdot G = r \cdot G + c \cdot x \cdot G$ which is an instance of the discrete logarithm problem. He could not cheat by enumerating random r values, as it would be too hard to find a matching value for c.

5.4. Zero knowledge test of discrete logarithm equality

Suppose that Prover knows two publicly known quantities that have the same discrete logarithm x to publicly known respective bases G and H of the group F_n.

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$ and $H \in E/F_n$. Prover claims he knows x such that $B = x \cdot G$ and $C = x \cdot H$ and wants to prove knowledge of this fact without revealing x. The procedure was originally proposed in [49], and its ECC analogous is as follows:

Protocol Steps:

- Prover chooses random $r \in F_n$ and computes the points $K = r \cdot G$ and $L = r \cdot H$
- Prover sends the points K, L to Verifier
- Verifier chooses random $c \in F_n$ and sends c to Prover
- Prover computes $m = r + c \cdot x (mod\, n)$ and sends m to Verifier
- Verifier checks that $m \cdot G = (r + c \cdot x) \cdot G = r \cdot G + c \cdot x \cdot G = K + c \cdot B$
- Verifier checks that $m \cdot H = (r + c \cdot x) \cdot H = r \cdot H + c \cdot x \cdot H = L + c \cdot C$

Why it works: In this protocol the Prover claims he knows x as the discrete logarithm of two public quantities B, C. His actions are similar with Schnorr's protocol but for the two public quantities. For example in the first step he computes 2 points on the curve K, L that will be used for the verification. It can also be made non-interactive by the applying the Fiat–Shamir trick: the Prover simulates the Verifier by computing c with a secure hash function as HASH(B, G, C, H, K, L).

5.5. Zero knowledge proof of single bit

Prover and Verifier agree on an elliptic curve E over a field F_n, a generator $G \in E/F_n$ and $H \in E/F_n$. Prover knows x and h such that $B = x \cdot G + h \cdot H$ where $h = \pm 1$. He wishes to convince Verifier that he really does know x and that h really is $\pm$ 1 without revealing x nor the sign bit [46].

Protocol Steps:

- Prover generates random $s, d, w \in F_n$
- Prover computes the points $A = s \cdot G - d \cdot (B + h \cdot H)$ and $C = w \cdot G$
- If $h = -1$ Prover swaps $A \leftrightarrow C$
- Prover sends the points A, C to Verifier
- Verifier generates random $c \in F_n$ and sends c to Prover
- Prover computes $e = c - d$ and $t = w + x \cdot e$ both $(mod\, n)$
- If $h = -1$ Prover swaps $d \leftrightarrow e$ and $s \leftrightarrow t$
- Prover sends to Verifier d, e, s, t
- Verifier checks that $e + d = c$, $s \cdot G = A + d \cdot (B + H)$ and that $t \cdot G = C + e \cdot (B - H)$

Why it works: It is straightforward to confirm that if B is really given by one of the two formulas the Prover claimed then Verifier's verification will succeed. It is also easy to see that the Prover does not give away any information that would allow the Verifier to deduce x nor the sign bit h. That is because x is hidden inside t after being multiplied with e and added in w. The sign bit h is randomized with the appropriate swaps in the case of -1.

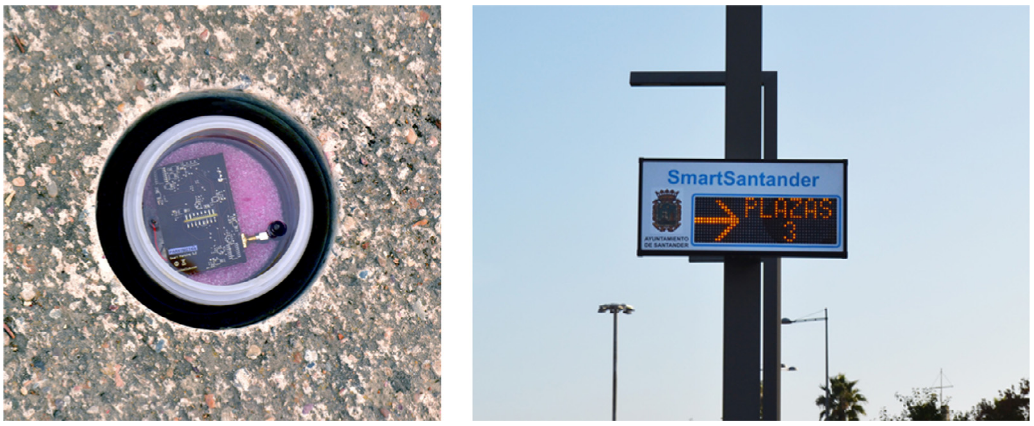

Fig. 1. Parking sensors buried under the road surface and Notification Displays to show free parking spots.

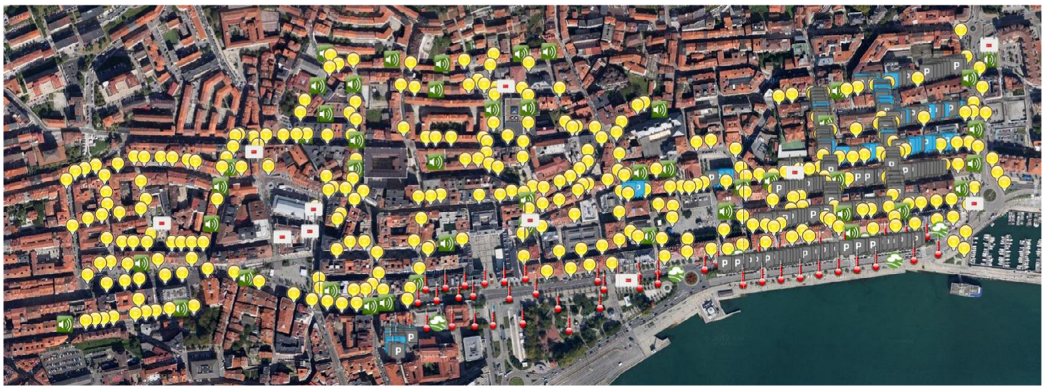

Fig. 2. Outdoor deployment of 400 parking sensors and 10 notification panels at the city of Santander.

6. Real-world performance evaluation

Designing an IoT system that can efficiently operate in a federated network of heterogeneous devices and network stacks, while being compliant with a plethora of privacy and trust requirements is a complex task. Simulations, as an important phase during the development of systems, are useful for developing further understanding the operating condition of a system. However, they suffer from several imperfections [50] as they make artificial assumptions on radio propagation, traffic, failure patterns, and topologies. What makes it particularly difficult is the strong dependency of IoT systems on real-world processes that are often a result of complex systems-of-systems interactions and extremely difficult to model accurately [51].

We strongly believe that delivering robust applications, requires testing and performance evaluation of individual system components as well as compositions of the system on real hardware in large-scale deployments. Given the increased production of IoT hardware nodes and the introduction of new tools for managing IoT testbeds (e.g., see [52]) a number of large scale outdoor testbed deployments have been established to help developers evaluate their technologies in real environments and with real end users (e.g.,see [53]).

One such environment is SmartSantander[12], a city scale testbed that supports experimentation using IoT technologies along with Smart City services provision, e.g., parking monitoring, environmental monitoring, precise irrigation in parks and augmented reality using NFC tags. The Outdoor Parking Management infrastructure deployed in the city of Santander supports the provision of a Limited Parking Space Management services. The existing deployment is based on almost 400 parking sensors (using ferromagnetic technology), buried under the asphalt in several hundred parking places in the city center of Santander. Information collected from the sensors is forwarded to the main storage (maintained at the cloud) where is then accessible to end-user applications (web/smartphone) that process the information for service provisioning (e.g., to guide drivers to free parking places). At central intersections of the main streets' of the city, 10 notification panels are installed in order to guide drivers towards the available free parking lots. Fig. 1 shows the two different types of hardware deployed. Deployment for outdoor parking area management is shown in Fig. 2. For a detailed description of the deployment and experimentation architecture of the IoT experimentation facility being deployed at Santander city see [53].

6.1. Santander smart city testbed facility

In more details, the city center is divided in 22 different zones. Each zone is assigned with a gateway device that is collecting data from the sensors and forwards them to the Internet through the GPRS interface. The parking sensors (IoT nodes) are provided with one 802.15.4 transceiver to send their parking state (free or occupied) to the corresponding gateway and one Digimesh transceiver for network management purposes (so that traffic for network management is not creating interference with traffic for service provisioning). The zones have different network parameters, creating independent networks that work on different frequency channels not to interfere with each other. Information received from the sensor nodes is stored and processed in the corresponding

[12] FP7-ICT-2009-5-257992. Project SmartSantander. http://smartsantander.eu

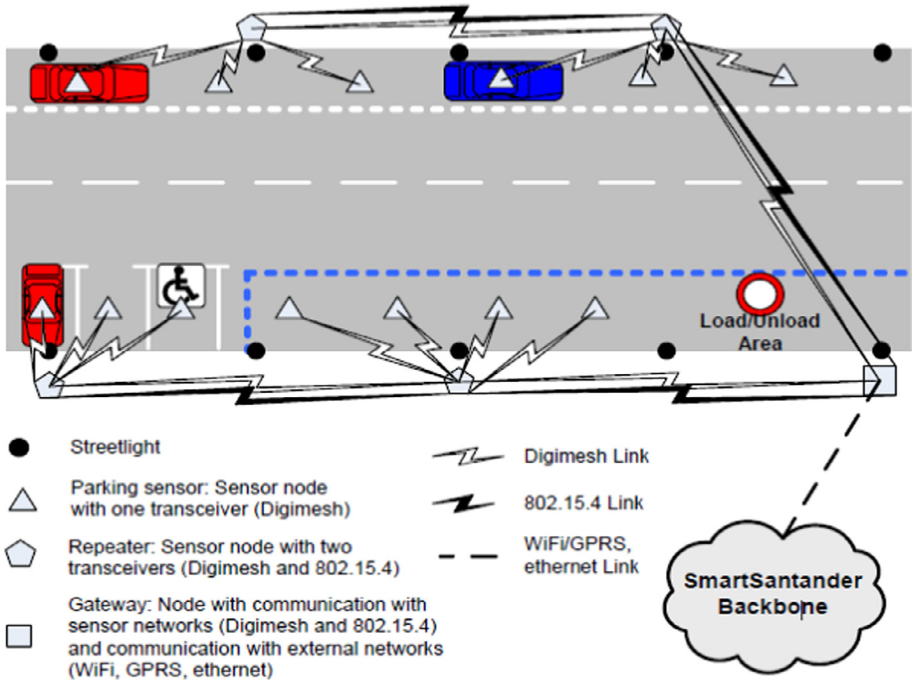
Streetlight
Parking sensor: Sensor node with one transceiver (Digimesh)
Repeater: Sensor node with two transceivers (Digimesh and 802.15.4)
Gateway: Node with communication with sensor networks (Digimesh and 802.15.4) and communication with external networks (WiFi, GPRS, ethernet)
Digimesh Link
802.15.4 Link
WiFi/GPRS, ethernet Link
SmartSantander Backbone
Load/Unload Area

Fig. 3. Architecture of Smart Parking infrastructure in the Santander deployment.

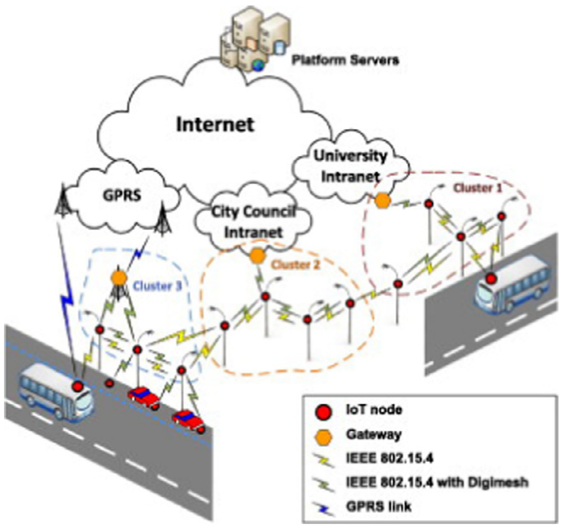

Fig. 4. Physical Network layout of Smart Parking infrastructure in the Santander deployment.

gateway, in order to be used by different applications running over it, both in a local way or accessing from Internet through the SmartSantander backbone. The network architecture is depicted in Fig. 3.

Securing the network infrastructure. By examining the physical network layout of the smart parking infrastructure in the Santander deployment (see Fig. 4) we observe a federation of mobile networks (GPRS), wireless networks (802.15.4, digimesh, wifi) and wireline. Some of these networks are operated by different organization, with different goals and orientation (e.g., the University intranet aiming for research & educational purposes, versus the telecom-operated mobile network aiming for commercial usages). It is evident that establishing a common security policy is a very complex task.

Securing the IoT nodes. Communication across the IoT domain is carried out using two 802.15.4 channels. The devices under the asphalt operate an ATMEGA1281 32-bit 14 MHz processor with 8KB SRAM/128KB FLASH while the gateway devices are 500 MHz x86-based debian systems with 256MB RAM/8GB storage space. Both devices support RFID/NFC/BLE communication with Smartphones (citizen's choice). Any solution beyond base AES/RSA encryption needs to be implemented for at least four different platforms and cross-layer key management schemes need to provided.

Securing the Cloud storage. Once location based information (and other sensitive personal information) is transmitted across the infrastructure will eventually reach the cloud layer leaving a data trail at different levels of the network (e.g., within a DTN bundle, at CDN level etc.). It is clear that a thorough examination is required to guarantee that private data is not stored or buffered across the system.

We believe that the Santander smart city testbed facility is a typical example of a heterogeneous deployment incorporating (a) a variety of hardware platforms (ranging from low-end 16 MHz devices, to cloud servers) and (b) a combination of different networking stacks and communication protocols (ranging from protocols for low-power lossy networks to ultra-fast broadband optical networks). The architecture of the Santander smart city IoT infrastructure is based on open standards and royalty-free software tools in order to avoid vendor-lock in future extensions of the infrastructure. It is therefore reasonable to expect new types of hardware and even completely new families of network protocols to be incorporated in the smart city facility. In this sense, it is imperative to provide solutions that are generic enough to guarantee cross-platform and cross-protocol operation.

6.2. Protocols implementation and evaluation

We now assess the performance of our approach in terms of execution time, code size and message size. Our privacy-preserving application is implemented based on the Wiselib algorithm library [42]. It is implemented in a platform-agnostic and OS-independent way and decomposed in a well defined set of algorithms that implement the required functionalities such as the ECDLP, the five zero-knowledge protocols discussed in Section 5 and the communication algorithms. Each sub-system follows the so-called "concept" design approach that defines common interfaces so that different protocols for the same *concept* are fully interchangeable. In particular our set of algorithms are implemented based on C++ and templates, but without virtual inheritance and exceptions. Algorithm implementations in **Wiselib** can be recompiled for several platforms and firmwares, like C (Contiki), C++ (iSense), and nesC (TinyOS), without the need to change the code. The library is also adapted for C-based mobile phone operating systems like Android (via JNI) and iPhone OS.

We evaluated the implemented system under different combinations of ZKP protocols for the two platforms used in the Santander smart parking deployment (equipped with ATMEGA1281 processor, and a 500 MHz linux-based system). In our evaluation we also included two additional low-end devices based on different microcontrollers (Jennic JN5139, TI MSP430) that are commonly used in WSN deployments. Since our implementations are based on Wiselib, the same code is able to execute on all these platforms by simply using the platform specific compiler.

The implementation of our Elliptic Curve Cryptography follows the library in [22]. This implementation defines a recommended elliptic curve [54] over binary fields with equation $y^2 + xy = x^3 + x^2 + 1$ along with the irreducible polynomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. The curve's order (the number of points on it, r) and the base point is $G(x, y)$ are listed on Table 1.

In our first set of experiments we assess the execution time of basic Elliptic Curve Operations on different hardware platforms see Fig. 2 (Table 2). As a hash function for the protocols that require one, we used the algorithm SHA-1 [55]. We notice that there has been no attempt to optimize the code in charge for the elliptic curve operations on specific hardware architectures. In this way, our solution implements a new cryptographic tool that is automatically compatible with all the platforms supported in the real-world deployment. One can observe that the elliptic curve scalar multiplication (Public Key Generation) is the most demanding arithmetic operation. It is also evident that the 500 MHz linux system is much faster in calculating all the examined operations than low-capabilities processors.

Anyway, all the considered devices can perform the basic elliptic curve operations in less than one minute. Considering that usually the parking time is much longer, the upper bound of one minute for such operations is acceptable to effectively complete the verification protocols as detailed in Table 5.

Our next set of experiments conduct a comparative study among the five ZKP protocols designed in Section 5. We have summarized the prover's (PRV) and verifier's (VER) actions for each of the implemented protocols in Table 3 specifying for each action the number of times it has to be performed.

Table 1
Parameters for the Basic Elliptic Curve Operations.

Parameter	Value
r	$0x40000000000000000000020108a2e0cc0d99f8a5ef$
x	$0x2fe13c0537bbc11acaa07d793de4e6d5e5c94eee8$
y	$0x289070fb05d38ff58321f2e800536d538ccdaa3d9$

Table 2
Execution Time of Basic Elliptic Curve Operations on different hardware platforms.

Operation	Execution time			
	JN5139	MSP430	ATMega1281	500 MHz linux
Private key generation	87 ms	0.3 s	0.18 s	1.03 ms
Public key generation (scalar multiplication)	11.121 s	58.02 s	35.49 s	1.91 s
Curve points addition	94 ms	0.29 s	0.17 s	1.12 ms
Private key addition (21 bytes)	5 ms	12 ms	7 ms	0.38 ms
Private key multiplication (21 bytes)	14 ms	2 ms	12 ms	0.72 ms
SHA-1 hash (250 bytes)	2 ms	31 ms	18 ms	0.03 ms

Table 3
Actions Held by the Prover (PRV) and Verifier (VER) for each protocol.

Protocol / Operation	Random key generation	Curve multiplication	Curve addition	SHA-1 hash	Private keys addition	Private keys multiplication	Msgs sent
ZKP of DL with coin flip PRV	1	1	–	–	1 (if tails)	–	2
ZKP of DL with coin flip VER	–	1	1 (if tails)	–	–	–	2
Schnorr's protocol PRV	1	1	–	–	1	1	2
Schnorr's protocol VER	–	2	1	2	–	–	2
Schnorr's signature protocol PRV	1	3	–	1	1	1	1
Schnorr's signature protocol VER	–	4	2	1	–	–	1
ZKP of DL equality PRV	1	2	–	–	1	1	2
ZKP of DL equality VER	1	4	2	–	–	–	2
ZKP of single bit PRV	3	3	2	–	2	1	2
ZKP of single bit VER	1	4	4	–	1	–	2

Table 4
Code Size of the Protocols for Prover the (PRV) and the Verifier (VER) on the Devices Used.

Protocol	Total code size (text + data + bss)		
	JN5139	MSP430	ATMega1281
ZKP of DL with coin flip PRV	7908 bytes	6517 bytes	6838 bytes
ZKP of DL with coin flip VER	7004 bytes	6289 bytes	6772 bytes
Schnorr's protocol PRV	7964 bytes	6759 bytes	7146 bytes
Schnorr's protocol VER	9292 bytes	9455 bytes	9562 bytes
Schnorr's signature protocol PRV	10584 bytes	10433 bytes	10562 bytes
Schnorr's signature protocol VER	9540 bytes	10053 bytes	10102 bytes
ZKP of DL equality PRV	8568 bytes	7697 bytes	8110 bytes
ZKP of DL equality VER	8964 bytes	8519 bytes	8718 bytes
ZKP of single bit PRV	9604 bytes	9471 bytes	9512 bytes
ZKP of single bit VER	9032 bytes	7455 bytes	8274 bytes

Table 6
Size of Messages Exchanged by the Prover (PRV) and the Verifier (VER) for each Protocol.

Protocol	Message size
ZKP of DL with coin flip PRV	Point Message: 43 bytes New Key Message: 22 bytes
ZKP of DL with coin flip VER	Coin Message: 2 bytes Final Message: 1 byte
Schnorr's protocol PRV	Point Message: 43 bytes New Key Message: 22 bytes
Schnorr's protocol VER	Hash Message: 22 bytes Final Message: 1 byte
Schnorr's signature protocol PRV	Point and Key Message: 149 bytes (in two pieces)
Schnorr's signature protocol VER	Final Message: 1 byte
ZKP of DL equality PRV	Points Message: 85 bytes New Key Message: 22 bytes
ZKP of DL equality VER	New Key Message: 22 bytes Final Message: 1 byte
ZKP of single bit PRV	Points Message: 85 bytes New Key Message: 85 bytes
ZKP of single bit VER	New Key Message: 22 bytes Final Message: 1 byte

Based on the above decomposition, we proceed by measuring the code size of the protocols on the considered devices (we here exclude the linux-based system). The compiled code actually fits fairly well (approximately 8Kb) on the tiny memory of their processors as shown in Table 4. The difference in the protocols code size is due to the employment of different compilers.

We now proceed by examining the total execution time of each ZKP protocol. Table 5 lists the time elapsed between the prover's beginning of the protocol until the verifier's final response of whether she accepts or rejects the proof. We can observe that an interactive protocol like the first one, which requires a large number of execution rounds for verification, is not suitable for low-constrained devices. We thus conclude that the usage of protocols that need one round of execution, as the rest of the protocols considered in the table, is advised. Similar observations are reported in [56].

Communication between the *User* and *Verifier* can be carried out using RFID/NFC or WIFI. Table 6 describes the messages exchanged between the prover (PRV) and verifier (VER) and their sizes, for the completion of each protocol. Message size is an important parameter when considering low power wireless communication protocols like 802.15.4. All of our protocols messages except one, fit well on a single 802.15.4 packet (128 bytes). This fact is remarkable, as bigger messages would result to more message exchanges which in turn would imply greater energy consumption and possibly wireless medium congestion. The only exception is the Schnorr's Non-Interactive Protocol PRV message that requires 149 bytes and thus need to be broken in two pieces.

6.3. *Discussion on the applicability of our approach*

Our evaluation points out that the use of ZKP protocols under the elliptic curve cryptography setting is beneficial. We provide a cross-platform cryptographic mechanism that uses small key sizes to achieve the same level of security as other public key cryptosystems. Our implementation supports different hardware platforms thus increasing the degrees of freedom of our smart parking application. In this section we discuss in details the drawbacks of using elliptic curve cryptography in resource constrained environments and in particular to a smart city application context.

Our design allows to interchange the ECC encryption method supporting different implementations, possibly improving the performance by exploiting platform-specific optimizations. As we have already mentioned we did not make any attempt to optimize the code of the ECC module. One can easily observe that the curve point multiplication which is the basic action for each protocol, needs the most time to execute (35 s on ATMega1281, 11 s on JN5139, 58 s on MSP430). We believe that this is the *price* we have to pay to write a portable code that can run on a number of heterogeneous platforms without modifications. Although, Wiselib offers the chance to compile your code exploiting some platform specific features (e.g., 32-bit processor or hardware speedups) we aimed at generality and portability. The goal of this work was not to achieve some faster platform specific code (e.g. by employing low-level assembly code) as in [57,58]. Still, our design offers an easy way to incorporate such improvements as future work.

We think that for such low-end microprocessors running at 16 MHz or 8 MHz the total execution time of our protocols (except of course ZKP of DL with Coin Flip) is reasonable given the timeframe of a typical visit at a parking spot. For example, 33 s on a 16 MHz microcontroller for a secure verification is not too much to wait. Also when considering memory limitations, we showed that the protocols' compiled code actually fits well on the restricted memory of the devices used. Still, having implemented our protocols in Wiselib, our code can be executed on C-based operating systems for mobile phones like Android (via JNI) and iPhone OS where the embedded hardware is much more powerful. Thus our implementation allows the *Prover's* agent to be hosted in the IoT device stored in the car or in the citizen's smartphone. Similarly,

Table 5
Total Execution Time of the Protocols on the Devices Used.

Protocol	Required rounds	Total execution time			
		JN5139	MSP430	ATMega1281	500 MHz linux
ZKP of DL with coin flip	100 or more	2277 s	11802 s	7219 s	408 s
Schnorr's protocol	1	33.894 s	172.77 s	105.77 s	6.18 s
Schnorr's signature protocol	1	78.645 s	396.57 s	239.30 s	13.91 s
ZKP of DL equality	1	68.596 s	346.2 s	209.73 s	10.47 s
ZKP of single bit	1	80.46 s	462.74 s	258.73 s	13.05 s

the *Verifier's* agent can be hosted either in the IoT device under the asphalt or at the zone gateway device.

Our solution does not need to interact with an authority for *securing the network infrastructure*. The local message exchanges conducted between the *Prover* and *Verifier* user agents do not reveal any confidential information. Someone overhearing the medium, is unable to repeat the verification process. Our performance evaluation on the five different ZKP protocols considered, indicates that the first proposed protocol (ZKP of DL with Coin Flip) requires a large number of execution rounds in order to successfully complete. It performs a large number of arithmetic operations and it involves many message exchanges. Although such protocol could be executed fairly well in non-embedded systems, when the target are embedded devices capable of wireless communications, it is not suitable because it is very demanding in terms of messages exchanged that can possibly congest the wireless medium. The rest of the protocols require only one round of execution [56], much smaller number of messages (2–4 messages) and complete much faster and with less energy consumption. Schnorr's protocol required the least time to execute – 105 s on the ATMega1281 microcontroller, which is quite fair considering the processing power and the memory of the device used.

An additional disadvantage of the first protocol is that it cannot be considered as secure as the rest when dealing with wireless communication. A malicious verifier or an adversary who eavesdrops the communication between the Prover and the Verifier could replay the proof to another party using the overheard data. That is because the verifier's responses consist of just a single bit (simulating a coin flip). In all the other protocols the Verifier responses involve fresh random data (e.g. using a hash function) and such an attack could not work.

Another issue concerns the comparison between Schnorr's and Schnorr's non-interactive protocol. By transforming Schnorr's protocol to a non-interactive protocol, the required message exchanges are reduced (1 message required by the Prover and Verifier). However, the transformed protocol is not as efficient as Schnorr's original protocol in terms of execution time, code size and message size.

As far as the protocols' message sizes are concerned, we observe that they are acceptable. Most of the messages exchanged are 43 or 85 bytes which can fit on a single 802.15.4 packet (max payload 128 bytes). Only, the Prover's message from Schnorr's non-interactive protocol was too large (149 bytes) to fit in a single packet and thus it has been broken in two pieces. We observe that, by recompiling each device's firmware, which is a straightforward procedure, one could increase the maximum payload size and such a message could thus fit in a single packet. Such a solution though, would be non-standard. The reasonable message sizes are due to the ECC approach. A reader can easily realize that using another cryptosystem (e.g. with 1024-bit keys) would result in larger message sizes that would necessarily require fragmentation and a consequent increase in energy consumption and medium congestion.

Finally, a point for discussion is the fact that in our protocols, we pre-loaded the elliptic curve used and its parameters on the devices memory. This way, the elliptic curve and its parameters could be compromised by physical and tampering attacks. However, we believe that in the future, most devices will be equipped with Trusted Platform Module (TPM), namely secure cryptoprocessors suitable for storing cryptographic keys and protecting private information from physical and tampering attacks.

7. Conclusions and future work

In this paper we follow a privacy-by-design approach and introduce a privacy-preserving smart parking application system. We argue that in city-scale IoT deployments applying a security scheme across the federated infrastructure is extremely challenging. Our approach totally avoids transmitting confidential information between the system agents. Therefore our application is suitable also in cases where untrusted networks are included in the federated infrastructure. We describe the operation of our system under three use case scenario and show the benefits of utilizing zero-knowledge proofs.

Considering the Santander smart city IoT deployment, we looked into the problem of implementing zero-knowledge protocols (ZKP) in low-end devices. Based on the resource limitations of such devices as well as the restrictions imposed by low power wireless communication protocols (e.g. IEEE 802.15.4) we applied the elliptic curve cryptography (ECC) approach. Specifically, we have carefully transformed well established zero-knowledge protocols based on the discrete logarithm problem (DLP) under the elliptic curve discrete logarithm problem (ECDLP). This transformation step was the key for implementing such heavy protocols, in terms of computation and communication, on constrained environments, due to the fact that ECC offers similar levels of security with other cryptosystems (e.g. RSA) using smaller keys. For the first time, we present an implementation of ZKP protocols in an open source and generic programming library called Wiselib. Our code is highly portable, freely available and ready to use as part of Wiselib. Based on our implementations, we conducted a thorough and comparative evaluation of the protocols on two hardware platforms used in the Santander smart city testbed and on two popular hardware platforms equipped with widely used low-end microcontrollers.

We are strongly confident that zero-knowledge proofs can be used as a privacy-preserving tool and our approach can be applied to other smart city applications consisting of different kinds of devices, e.g., for intelligent transportation systems, smart campuses etc. We highlight the advantages and point out the disadvantages of our approach and provide some valuable technical insights related to IoT hardware and networking technologies. We also need to emphasize that ECC is still public key cryptography (just with smaller key parameters etc.) – it's just much more efficient than the conventional PKC like RSA etc. Our work shows that there is still an efficiency issue to be considered but we should consider that at the same time the platforms' hardware is improving. Remark that our implementation is as generic as possible and this means that we do not use any software optimizations (which would introduce larger code size and memory overhead) or inline assembly code (using `#ifdefs`) which could speed up things a lot – but only for specific platforms. Such optimizations and getting the best efficiency out of ECC were not the goal of this work. In contrast our goal was to provide real-world evidence on the feasibility of a generic approach for constrained settings of an IoT infrastructure used in current smart city deployments. We believe that our results can be used by developers that wish to provide certain levels of security and privacy in their applications.

Future work includes the expansion of the library presented in this paper so as to include ZKIP protocols which prove various relations for the encoded values without revealing them (e.g. prove that my age is over 18 years without revealing it). This way, our library can be the basis for implementing attribute based credentials [59,60] on embedded devices.

References

[1] P.A. Pavlou, State of the information privacy literature: where are we now and where should we go? MIS Q. 35 (4) (2011) 977–988.
[2] B.A. Price, K. Adam, B. Nuseibeh, Keeping ubiquitous computing to yourself: a practical model for user control of privacy, Int. J. Hum. Comput. Stud. 63 (1) (2005) 228–253.
[3] F. Bélanger, R.E. Crossler, Privacy in the digital age: a review of information privacy research in information systems, MIS Q. 35 (4) (2011) 1017–1042.

[4] D. Christin, A. Reinhardt, S.S. Kanhere, M. Hollick, A survey on privacy in mobile participatory sensing applications, J. Syst. Softw. 84 (11) (2011) 1928–1946.
[5] T. Baumgartner, I. Chatzigiannakis, S.P. Fekete, S. Fischer, C. Koninis, A. Kröller, D. Krüger, G. Mylonas, D. Pfisterer, Distributed algorithm engineering for networks of tiny artifacts, Comput. Sci. Rev. 5 (1) (2011) 85–102, doi:10.1016/j.cosrev.2010.09.006.
[6] D.J. Malan, M. Welsh, M.D. Smith, A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography, in: Proceedings of the First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, SECON 2004, October 4–7, 2004, Santa Clara, CA, USA, 2004, pp. 71–80, doi:10.1109/SAHCN.2004.1381904.
[7] I. Chatzigiannakis, E. Konstantinou, V. Liagkou, P.G. Spirakis, Design, analysis and performance evaluation of group key establishment in wireless sensor networks, Electr. Notes Theor. Comput. Sci. 171 (1) (2007) 17–31, doi:10.1016/j.entcs.2006.11.007.
[8] K. Maletsky, Atmel white paper: Rsa vs ecc comparison for embedded systems, 2015, atmel-8951A-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper_07201.
[9] J. Marceau, Introduction: innovation in the city and innovative cities., Innov.: Manag. Policy Pract. 10 (2008).
[10] H. Wang, W. He, A reservation-based smart parking system, in: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, IEEE, 2011, pp. 690–695.
[11] Z. Pala, N. Inanc, Smart parking applications using rfid technology, in: RFID Eurasia, 2007 1st Annual, 2007, pp. 1–3, doi:10.1109/RFIDEURASIA.2007.4368108.
[12] J. Krumm, A survey of computational location privacy, Pers. Ubiquitous Comput. 13 (6) (2009) 391–399.
[13] W. Diffie, M.E. Hellman, New directions in cryptography, Inf. Theory IEEE Trans. 22 (6) (1976) 644–654.
[14] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, Comparing elliptic curve cryptography and rsa on 8-bit cpus, in: Cryptographic hardware and embedded systems-CHES 2004, Springer, 2004, pp. 119–132.
[15] H. Wang, W. He, A reservation-based smart parking system, in: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, 2011, pp. 690–695, doi:10.1109/INFCOMW.2011.5928901.
[16] N. Hanif, M. Badiozaman, H. Daud, Smart parking reservation system using short message services (sms), in: Intelligent and Advanced Systems (ICIAS), 2010 International Conference on, 2010, pp. 1–5, doi:10.1109/ICIAS.2010.5716179.
[17] S. Kim, D. Kim, Y. Ryu, G. Kim, A robust license-plate extraction method under complex image conditions, in: Pattern Recognition, 2002. Proceedings. 16th International Conference on, vol. 3, 2002, pp. 216–219 vol.3, doi:10.1109/ICPR.2002.1047833.
[18] K.T. Nguyen, M. Laurent, N. Oualha, Surveyon secure communication protocols for the internet of things, Ad Hoc Netw. 32 (C) (2015) 17–31, doi:10.1016/j.adhoc.2015.01.006.
[19] H. Ahmadi, N. Pham, R. Ganti, T. Abdelzaher, S. Nath, J. Han, Privacy-aware regression modelling of participatory sensing data, in: SenSys'10: The ACM Conference on Embedded Networked Sensor Systems, Association for Computing Machinery, Inc., 2010.
[20] L.B. Oliveira, R. Dahab, J. Lopez, F. Daguano, A.A. Loureiro, Identity-based encryption for sensor networks, in: Pervasive Computing and Communications Workshops, 2007. PerCom Workshops' 07. Fifth Annual IEEE International Conference on, IEEE, 2007, pp. 290–294.
[21] G. Gaubatz, J.-P. Kaps, B. Sunar, Public key cryptography in sensor networks–revisited, in: Security in Ad-hoc and Sensor Networks, Springer, 2005, pp. 2–18.
[22] D.J. Malan, M. Welsh, M.D. Smith, Implementing public-key infrastructure for sensor networks, TOSN 4 (4) (2008).
[23] G. Piro, G. Boggia, L.A. Grieco, A standard compliant security framework for ieee 802.15. 4 networks, in: Internet of Things (WF-IoT), 2014 IEEE World Forum on, IEEE, 2014, pp. 27–30.
[24] J. Zhang, V. Varadharajan, Wireless sensor network key management survey and taxonomy, J. Netw. Comput. Appl. 33 (2) (2010) 63–75. http://dx.doi.org/10.1016/j.jnca.2009.10.001.
[25] H. Chan, A. Perrig, Security and privacy in sensor networks, Computer 36 (10) (2003) 103–105.
[26] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330. http://dx.doi.org/10.1016/j.comnet.2008.04.002.
[27] N. Li, N. Zhang, S.K. Das, B. Thuraisingham, Privacy preservation in wireless sensor networks: a state-of-the-art survey, Ad Hoc Netw. 7 (8) (2009) 1501–1514. Privacy and Security in Wireless Sensor and Ad Hoc Networks. http://dx.doi.org/10.1016/j.adhoc.2009.04.009.
[28] S. Sicari, A. Rizzardi, L. Grieco, A. Coen-Porisini, Security, privacy and trust in internet of things: the road ahead, Comput. Netw. 76 (2015) 146–164.
[29] CGAL: Computational Geometry Algorithms Library. http://www.cgal.org.
[30] Boost. http://www.boost.org.
[31] A. Dunkels, B. Gronvall, T. Voigt, Contiki - a lightweight and flexible operating system for tiny networked sensors, in: LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, 2004.
[32] TinyOS, (http://www.tinyos.net).
[33] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, D. Culler, The nesc language: A holistic approach to networked embedded systems, in: Proceedings of Programming Language Design and Implementation (PLDI) 2003, 2003.
[34] Z. He, F. Österlind, A. Dunkels, An adaptive communication architecture for wireless sensor networks, in: Proceedings of ACM SenSys '07, 2007.
[35] A. Dunkels, Poster abstract: Rime – a lightweight layered communication stack for sensor networks, in: Proceedings of EWSN 2007, Poster/Demo session, 2007.
[36] R. Sauter, P.J. Marrón, A. Dunkels, T. Voigt, N. Tsiftes, N. Finne, F. Österlind, J. Eriksson, Demo abstract: Towards interoperability testing for wireless sensor networks with cooja/mspsim, in: Proceedings EWSN '09, 2009.
[37] O. Akribopoulos, V. Georgitzikis, A. Protopapa, I. Chatzigiannakis, Building a platform-agnostic wireless network of interconnected smart objects, in: 15th Panhellenic Conference on Informatics, PCI 2011, Kastoria, Greece, September 30 - October 2, 2011, 2011, pp. 277–281, doi:10.1109/PCI.2011.58.
[38] A. Boulis, C.-C. Han, M.B. Srivastava, Design and implementation of a framework for efficient and programmable sensor networks, in: Proceedings of MobiSys '03, ACM, New York, NY, USA, 2003, pp. 187–200. http://doi.acm.org/10.1145/1066116.1066121.
[39] P. Levis, D. Culler, Mate: A tiny virtual machine for sensor networks, in: International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, USA, 2002.
[40] N. Shaylor, D.N. Simon, W.R. Bush, A java virtual machine architecture for very small devices, in: LCTES '03: Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems, 2003, doi:10.1145/780732.780738.
[41] D. Simon, C. Cifuentes, The squawk virtual machine: Java on the bare metal, in: OOPSLA '05, ACM, New York, NY, USA, 2005, pp. 150–151, doi:10.1145/1094855.1094908.
[42] T. Baumgartner, I. Chatzigiannakis, S.P. Fekete, C. Koninis, A. Kröller, A. Pyrgelis, Wiselib: A generic algorithm library for heterogeneous sensor networks, in: EWSN, 2010, pp. 162–177.
[43] A.J. Menezes, S.A. Vanstone, P.C.V. Oorschot, Handbook of Applied Cryptography, CRC Press, Inc., Boca Raton, FL, USA, 1996.
[44] S. Almuhammadi, N.T. Sui, D. McLeod, Better privacy and security in e–commerce: Using elliptic curve-based zero-knowledge proofs, in: CEC, 2004, pp. 299–302.
[45] C.P. Schnorr, Efficient signature generation by smart cards, J. Cryptol. 4 (1991) 161–174, doi:10.1007/BF00196725.
[46] W. Smith, Cryptography meets voting, 2005,
[47] D. Chaum, J.-H. Evertse, J. van de Graaf, An improved protocol for demonstrating possession of discrete logarithms and some generalizations, in: EUROCRYPT, 1987, pp. 127–141.
[48] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in: Proceedings on Advances in cryptology—CRYPTO '86, Springer-Verlag, London, UK, 1987, pp. 186–194.
[49] J. Boyar, D. Chaum, I. Damgård, T.P. Pedersen, Convertible undeniable signatures, in: CRYPTO, 1990, pp. 189–205.
[50] E. Egea-Lopez, J. Vales-Alonso, A.S. Martinez-Sala, P. Pavon-Marino, J. García-Haro, Simulation tools for wireless sensor networks, in: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS05), 2005, p. 24.
[51] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, T. Razafindralambo, A survey on facilities for experimental internet of things research, Commun. Mag. IEEE 49 (11) (2011) 58–67.
[52] G. Coulson, B. Porter, I. Chatzigiannakis, C. Koninis, S. Fischer, D. Pfisterer, D. Bimschas, T. Braun, P. Hurni, M. Anwander, G. Wagenknecht, S.P. Fekete, A. Kröller, T. Baumgartner, Flexible experimentation in wireless sensor networks, Commun. ACM 55 (1) (2012) 82–90, doi:10.1145/2063176.2063198.
[53] L. Sanchez, L. Muoz, J.A. Galache, P. Sotres, J.R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, D. Pfisterer, Smartsantander: iot experimentation over a smart city testbed, Comput. Netw. 61 (0) (2014) 217–238. Special issue on Future Internet Testbeds Part I. http://dx.doi.org/10.1016/j.bjp.2013.12.020.
[54] Certicom research : Sec 2 - recommended elliptic curve domain parameters, 2010, (http://www.secg.org/collateral/sec2_final.pdf).
[55] D. Eastlake, P. Jones, US Secure Hash Algorithm 1 (SHA1), Technical Report 3174, 2001.
[56] S. Almuhammadi, C. Neuman, Security and privacy using one-round zero–knowledge proofs, in: CEC, 2005, pp. 435–438.
[57] N. Gura, A. Patel, A. Wander, H. Eberle, S.C. Shantz, in: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs, 2004, pp. 119–132.
[58] A. Liu, P. Ning, Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks, in: IPSN, 2008, pp. 245–256.
[59] J. Camenisch, E.V. Herreweghen, Design and implementation of the idemix anonymous credential system, in: ACM Conference on Computer and Communications Security, 2002, pp. 21–30.
[60] S.A. Brands, Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy, MIT Press, Cambridge, MA, USA, 2000.