Chapter 1

Securing Communications Among Severely Constrained, Wireless Embedded Devices

Alexandros Fragkiadakis, George Oikonomou, Henrich C. Pöhls, Elias Z. Tragos, Marcin Wójcik

1.1 Summary

The goal of this chapter is to present the ideas and concepts of the EU-FP7 SMARTCITIES project "RERUM" with regards to improving the communication security in IoT-based smart city applications. The chapter tries to identify the gaps in previous IoT frameworks with regards to security and privacy and shows the advances that RERUM brings to the IoT community with its significant focus on embedded device functionalities. The goal of the RERUM secure communications framework is to provide light-weight solutions so that they can be applied even in the very constrained IoT devices. Solutions for lightweight encryption (based on the relatively new theory of Compressive Sensing), on transport-layer security (based on DTLS) and on integrity verification of data (using on-device signatures) are presented in detail, discussing their applicability and the benefits they bring to IoT.

1.2 Introduction

The Internet of Things (IoT) is considered to be an important factor of economic growth. This massive network of heterogeneous devices (or interchangeably: machines, things, and objects) is becoming part of peoples everyday lives and will continue penetrating our day-to-day activities, as smart devices become ubiquitous. This idea of "ubiquitous computing" stems from Mark Weiser, who in 1991 presented several use cases which are still open today and for his time they were considered to be "science fiction" [1]. Use cases like the alarm clock that asks the user if she wants coffee and then the coffee maker prepares it or the window that shows the weather forecast for today are only nowadays considered possible with the latest advances of the IoT technologies.

Due to the wide range of possible applications that it can support, the IoT has gained much research attention the last few years. Everyday objects (e.g. chairs, windows, fridges, books, tables, cars, houses, trees) are interconnected, communicate with each other, exchange the information they sense, and thus become "smart"

equipped with intelligence. However, users and service providers are reluctant to exploit this IoT potential without assurance for the safety of their private information. Why would someone spend money to buy and install an IoT-based home automation system, if his neighbours or any potential burglars would be able to identify when he is at home, what type of devices he has or hack his door and enter the house without any obtrusion? Why would someone use a crowdsourcing application for traffic monitoring if the system would be able to track him and know his location at any given time? Why would someone use smart health products for monitoring his vitals if he is not sure that his health-related information would not be sent to his insurance company to influence the amount of money he pays for his insurance?

With the number of interconnected smart devices increasing exponentially (reports estimate that 50 billion devices will be connected to the Internet by 2020 [2]) new security and privacy issues arise, regarding i.e. confidentiality, data integrity, information privacy, and safety. Although there is a lot of work in the literature for handling these issues in standard communication networks and systems, it is not easy to apply existing techniques to IoT systems. The main issue arises from the fact that the IoT does not include only mobile phones, standard laptops, PCs, etc. like standard communication systems, but it includes mainly devices that are constrained in terms of computational power, memory, and energy, like standard wireless sensor platforms. Thus, existing sophisticated and standard security/privacy solutions cannot be easily applied on IoT systems with constrained devices.

Security and privacy in the IoT have only recently received a lot of attention, as the technologies become mature to be commercialized. However, The complexity of existing security/privacy solutions for the standard Internet and the resource constraints of the IoT devices are the main reasons for not embedding any type of security/privacy mechanisms on the devices (until recently). Many recent reports indicate a clear lack of security mechanisms for the communication of smart devices that are utilized currently by IoT applications [3, 4, 5]. The lack of security mechanisms can result to a number of threats for the users, mostly related with the privacy of their personal information (e.g. health information, name, location, even credit card numbers), the provision of unauthorised access to the systems of the users (even their houses if they are managed by home automation systems) or in extreme situations to affect the health of the users (e.g. if the window actuator malfunctions and closes suddenly hitting the user). From the latter example it can be seen that security threats in IoT systems can be initiated not only by malicious users, but also from malfunctioning devices.

Thus, in addition to the need to deploy existing sophisticated standard security and privacy solutions in the unconstrained world, the constrained nature of IoT poses itself an additional challenge. This chapter aims to give some ideas and to propose a basic framework for communication security in IoT, focusing on solutions that can be easily implemented on constrained embedded devices.

1.3 Related Work

To ensure that an IoT system protects the user's data and the user safety, a number of cross-layer security, privacy and trust mechanisms have to be taken into account from the design phase [6]. Furthermore, end-to-end security and privacy is needed to ensure strong data protection across all the involved technologies and communication systems that are used to transfer the information from the device to the application [7].

Indeed, due to the fact that IoT systems can involve a number of heterogeneous technologies, the communication layer is one of the main elements that have to be protected from attacks. As it is described in [8], attacks in the communication channel are quite severe because they can result to data loss (either due to interference/jamming or intermediate nodes dropping packets), tampering of data (altered by intermediate malicious hops) or replay attacks. All of these affect the integrity of the data that are used in the IoT applications and can degrade the trustworthiness and the reliability of the overall system.

As explained before, not much work has been done in the areas of communication security and privacy in IoT. This is also proved by the focus of the previous EU-funded IoT projects, which was mainly in the areas of interconnecting systems, analysing and processing data and exposing services. IoT-A [8] had defined a *Security Functionality Group* that ensures the security and privacy of an IoT system, but with a focus on authentication, authorisation, identity and key management and trust. Regarding secure communications, IoT-A focused on a framework for key exchange and management to allow the secure distribution of keys between devices. In i-Core [9] the security-related focus was on enhanced access control using context-awareness and not on communication security or on embedded security on IoT devices. BUTLER [10] had overall a more device-oriented focus compared with other IoT projects, but it's communication security focus was limited on authorization and device authentication. IoT@Work [11] had a cross-layer security component which included also device integrity assurance and network access control, applying also some configuration on network devices. OPENIOT [12] also focused only on service-based authorisation, authentication, access control and identity management. COSMOS [13] is working on hardware embedded security functionalities, however, the proposed mechanisms are not lightweight for standard IoT devices and thus the selected hardware device is a relatively powerful XILINX ZC702 [14]. SMARTIE [15] is also working on authentication and authorization and on device-oriented topics such as secure storage, but without supporting DTLS for securing the CoAP communications.

It is evident from the above, that the main focus of the IoT community was limited on the service-oriented security functionalities for authentication, authorization and access control. The target of RERUM is to try to secure also the "last mile", namely the communication between the IoT gateways and the constrained devices. The key functionalities presented below are (i) Compressive Sensing based encryption, (ii) DTLS-based secure communication and (iii) integrity protection using on device signatures.

Several contributions have studied CS encryption strength, most of them focusing on the computation secrecy of this scheme. The authors in [16] investigate CS encryption strength for two different type of attacks. For the brute-force case, they show that the complexity of this attack is in the order of $O(N(1/2))$. For the sparsity-related attack, an attacker has to check all possible column combinations of the measurement matrix, something that requires extensive resources, making this attack infeasible. In [17], the authors show that when an attacker decrypts the ciphertext using a wrong measurement matrix, then, the sparsity over the decrypted plaintext is higher than the sparsity of the original one. A multi-class CS encryption scheme is described in [18] where legitimate receivers gain multi-level confidential access based using different measurement matrices.

With the advent of elliptic curve based cryptography (ECC) it was possible to have constrained devices generate signatures, which was not really feasible with RSA based schemes. ECC was presented already in 1986 independently by *Miller* [19] and *Koblitz* [20]. Implementation for wireless sensor networks exist, e.g., TinyECC [21], NanoECC [22], or NIST's ECClight [23]; and they have been further optimised [24]. Besides *Ayuso* et al.'s work [24], the 160bit curve implementation by *Kern* and *Feldhofer* [25] or the very lightweight ECC based construction for authentication of *Braun*, *Hess* and *Mayer* [26] on RFID-type devices are all examples showing that ECC is a very good candidate for constrained devices. As the ECC curves previously standardised by NIST have been criticised [27] new curves are en route to standardisation, for example the signature algorithm named ed25519 [28]. In general, ECC has shown to be usable in software implementations, and is well supported in today's recent IoT hardware platforms [29, 30, 31].

1.4 Secure Communications for the IoT

In this chapter, the focus is on the communication layer of IoT. We present a framework that can be used for enhancing the communication security and is especially built upon the requirements of networks of constrained devices. We have to note, that in this work, we assume that security includes the concepts of confidentiality and integrity (including authentication and origin) of the information exchanged between constrained IoT devices over a wireless network (of any technology). We start by describing from a forensic examiner's perspective the security issues of the constrained devices and the resulting problems on the IoT applications. Then we present three main communication security techniques as they have been developed in the FP7-SMARTCITIES project RERUM [32, 6]: (i) a lightweight encryption technique using the Compressive Sensing (CS) theory [33], (ii) the integrity protection of the data by signing the data to be transmitted from the constrained devices using digital signatures, and (iii) a security mechanism using the Datagram Transport Layer Security (DTLS) protocol.

The proposed techniques can work either as standalone, separately from each other, or as an integrated communication security framework providing maximum security and protection.

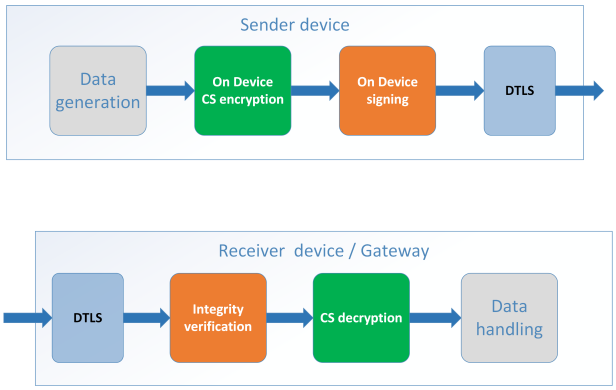

Figure 1.1: Communication security framework.

In Figure 1.1 integration of these techniques and the order they can be applied for the transmission of data from one device to another (or the gateway) is depicted. The figure is split into layers; the upper part shows the sender device and the lower part shows the receiver. We assume to have a scenario where the IoT device includes one (or more) sensors that gather measurements monitoring the attributes of a physical entity (in the IoT-A terminology [8]), meaning e.g. the temperature of a room, the status of a window, the energy consumption of the fridge, etc.

Let's assume that in our case, the scenario includes a device with a sensor that monitors the temperature of the living room in a household and is used for managing the air conditioning system. The device transmits periodically measurements to the gateway. Several attacks on the integrity of the temperature measurements can take place in such scenario, e.g. a malicious user can (i) alter the readings, sending higher temperature values, so that the air conditioning tries to make the room colder or (ii) intercept the temperature values trying to identify the presence of the inhabitant in the room (e.g. when there is a person in the room the temperature of the room may rise).

The proposed framework can mitigate these attacks by introducing confidentiality and integrity mechanisms on the constrained devices. As depicted in Figure 1.1, the process that is followed for the transmission of the data starts with the encryption of the data using the Compressive Sensing (CS) theory. This technique achieves simultaneously encryption and compression of the data, contributing also to the extension of the lifetime of the IoT devices due to the fact that they perform a much smaller number of transmissions, thus consuming less energy. Then, after the encryption, the packets to be transmitted are signed using i.e. classical signatures like ECDSA [34], group signatures [35] or malleable signatures [36]. This will provide authentication and integrity protection, ensuring that any modification of the measurements by unauthorised intermediate nodes will be identified and the modified data will be discarded and not allowed to affect the system's decisions. Finally, the data are being handled by DTLS, which is a cryptographic protocol protecting both the integrity and the confidentiality of the data. Using DTLS again on data that

is already encrypted and signed might sound like a repetition, but one shall not that DTLS protects communication channels, and CS and signatures protect the messages regardless if they are in-transit over a communication channel or at rest.

At the receiver side, the opposite order is followed and after DTLS the integrity of the data is verified and if this step is successful then the CS decryption module decrypts and decompresses the data forwarding them to the data handling module (which can be either a service or the routing protocol to forward the data to the IoT middleware, the applications or another device). Of course the order of these techniques is not fixed and it is left as a design choice of the system administrator, meaning that e.g. the measurements can be first signed one by one and then encrypted and transmitted using DTLS. The order of the application of the techniques does not normally affect the resulting improvement in the security of the framework.

In Figure 1.2 the application of these techniques on the various components of an IoT infrastructure is presented. As it can be seen, the one end of these mechanisms exists on the constrained IoT devices. This means that the encryption of the data using Compressive Sensing (CS), the signing of the data and a part of DTLS are lightweight enough to be running on the constrained devices. Now, the receiving part of DTLS runs mostly on the Gateway (or the destination IoT device in a single hop transmission), although it can also be an end-to-end protocol.

The decryption of the CS-based measurements can be done in various places, according to the design choice of the system administrator. The optimal solution would be to have it on the gateway (which is consider trusted) so that an adaptive CS-based framework (as the one proposed in [37] can be applied easily without increasing the overall backbone signalling. However, the decryption can also be done in the middleware if the administrator wishes to minimize the complexity of the gateway. In an extreme scenario, the decryption can also be done on the application in order to maximize the protection of the user data that are transmitted (so that no component of the system can identify these data).

Similarly, for the integrity verification, several different options are possible. It can be performed on the gateway, to ensure that all modified data will be discarded at the earliest possible step and improving the reliability of the system, avoiding modified data to be transmitted in the backbone system. Furthermore, the verification can be done on the middleware for reducing the complexity of the gateway. Finally, if the administrator wishes to have end-to-end data integrity protection, the verification can be indeed done also on the applications.

A more thorough analysis of the techniques follows in the next sections of this chapter, discussing in detail the IoT-specific problems they try to solve and how they succeed it being embedded on the constrained IoT devices. However, in order to give an in depth analysis of why it is important to have embedded security on the devices, the next section provides some results of digital investigations for networks of severely constrained embedded devices, bringing out some of the vulnerabilities of those networks providing insight on the amount of information that is stored and can be retrieved from devices of this nature by using relatively simple methods and tools.

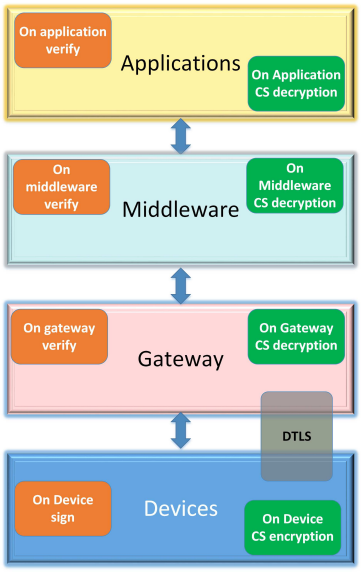

Figure 1.2: Mapping of the communication security techniques on the IoT infrastructure.

1.5 Digital Investigations for the Internet of Things

Following the discussion in the previous paragraphs and as the number of smart devices keeps increasing, so does the likelihood of malicious activity targeting those devices. Should an attack take place, post-incident forensic analysis can reveal information about the state of the device and network at the time of the attack and ultimately provide clues about the tools used to implement it, or about the attacker's identity. Law Enforcement Agencies (LEAs) currently do not have this capability, which is the topic of ongoing academic research [38].

Here, we focus on TCP/IP-based networks of severely constrained wireless embedded devices. IPv6 over Low Power Wireless Personal Area Networks (6LoWPANs) [39] and related Internet Engineering Task Force (IETF) specifications [40, 41] have made it possible to use IPv6 in networks of this nature. For those deployments, the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [42] is the de-facto standard for routing.

The adoption of protocols of the TCP/IP family has made those networks vulnerable to new security threats. For instance, the Internet Control Message Protocol Version 6 (ICMPv6) is used to perform key functions in IPv6 networks, one of which is Neighbor Discovery (ND). IPv6 ND's security vulnerabilities have been previously documented [43, 44], while a detailed overview of 6LoWPAN security threats and countermeasures is presented in [45].

Additionally, RPL itself has vulnerabilities that have been discussed by the research and standardisation communities: It is susceptible to selective forwarding,

wormhole and sinkhole attacks [46, 47, 48, 49], which can result in loss of data integrity, availability and confidentiality. The RPL specification defines some security counter-measures aiming to achieve confidentiality, integrity and replay protection [42]. Possible solutions have also been contributed by the academic community, such as VeRA [46]. However, these mechanisms are not widely adopted yet. For instance, the RPL implementation in the Contiki open source operating system for the IoT does not support any of them. Among the possible causes are problems with RPL's complexity and implementability, some of which have been previously documented [50]. Additionally, some of RPL's security services rely on the Advanced Encryption Standard (AES). Due to processing constraints of nodes forming 6LoWPANs, there is no well-established method for key negotiation and agreement. Recent research has demonstrated the feasibility of Elliptic Curve-based approaches [51].

To further clarify the motivation for digital investigation capability for 6LoWPANs, we use as an example the GINSENG research project. Its outputs demonstrated that it is possible to use a WSN of Contiki-powered nodes to control mission-critical applications in an operational oil refinery [52]. Even though this effort used a bespoke network stack, the control systems industry has been considering a move to open standards, such as TCP/IP and web technologies [53]. Thus, future adoption of 6LoWPAN for industrial automation is not inconceivable. This technological advance opens up the likelihood of injury or loss of life due to device malfunction or malicious activity. If this were to occur, investigating LEAs should be capable to discover forensic evidence in order to link an attack with the perpetrator.

Conceptually, post-incident analysis of an attack against a 6LoWPAN can be broken down into a number of complementary activities:

1. Analysis of devices forming the network,
2. Analysis of the network's traffic,
3. Analysis of log files

For item 1, we have already published some preliminary results in [38]. Briefly, in that work we presented an approach for the extraction and automated analysis of RAM and flash contents from a constrained wireless sensor node. Based on RAM and flash contents, we demonstrated that it is possible to automatically retrieve network-related information, such as routing table and ND cache contents. Moreover, we demonstrated the capability to correlate information gathered from multiple devices, in order to partially reconstruct the network topology at the time of extraction. The work focused on devices powered by 8051-based, 8-bit micro-controllers running the Contiki open source Operating System for the Internet of Things. However, it also includes an extensive discussion of how it can be extended in order to support different micro-controller architectures and different operating systems.

For item 2, our ongoing work focuses on an automated method for the post-hoc incident detection and analysis of specific 6LoWPAN-specific attacks through an examination of 6LoWPAN traffic captures. Previous work on forensic analysis of

WSN traffic has predominantly relied on powerful observer nodes forming part of a WSN deployment. For instance, a network of investigator nodes has been proposed as a solution for digital investigations of wormhole attacks [54]. Those observers are responsible for capturing sensor node behaviour and of forwarding this information to the network's base station. The same work proposes a set of algorithms to analyse evidence, in order to identify collaborating malicious nodes and to reconstruct wormhole attack scenarios. In a similar fashion, it has been demonstrated that a digital forensic readiness layer can be added over a pre-deployed IEEE802.15.4 WSN [55], by the addition of powerful forensic nodes that capture all WSN data plane traffic and maintain frame authenticity and integrity. This work mainly focuses on the reduction of time and cost involved in performing a digital investigation and demonstrates the ability to collect evidence without any modification to an existing network. Powerful observer nodes with the ability to analyse traffic and detect attacks have been adopted by the work documented in [56]. Observers can detect various patterns, such as wormhole, black-hole, sinkhole and sybil attacks. Only illegitimate behaviour is forwarded to the network's base station, thus reducing communication overheads. Foren6 is a recent research effort aiming to provide diagnostic and debugging capability for 6LoWPANs [57]. It is a passive monitoring tool capable of collecting information from multiple, potentially mobile sniffers. Foren6 stores a history of network state and topology changes, called versions, and provides the ability to navigate through the entire history in a post-hoc fashion through a network visualiser.

A traffic analysis tool that can identify attacks against RPL and ND in 6LoWPANs, flag events of interest and present results it to an investigator in a human-friendly format can be quite helpful. By combining an analysis of network activity with an analysis of the RAM contents of network nodes, one can reveal useful information about the events that led to a security incident.

Investigations can be further facilitated by a logging infrastructure (item 3). It is possible to log some information on devices themselves, but this has drawbacks: Limited on-device storage capacity means the logs can not be very detailed. Furthermore, logs will be distributed across the entire deployment and correlation will need to take place. It is also possible to implement a centralised logging infrastructure, for instance by using a syslog server. In this scenario, the drawback is that logging messages would increase network traffic and the approach would not scale well with deployment size and level of required logging detail. Investigations of hybrid approaches, whereby logs are cached locally and sent to a remote location periodically, possibly in an aggregated fashion, have still not been performed and can potentially give very good results.

1.6 Compressive Sensing Encryption

Usually, privacy and security become feasible through the use of encryption for the data exchanged between the communicating parties. Several algorithms based on public key encryption involving public and private keys (e.g. RSA [58]), provide robust encryption against unauthorised users. However, this type of algorithms re-

quire advanced resources, in terms of processing power and memory; hence, cannot be easily used by the resource-constrained sensors. On the other hand, symmetric algorithms, like the AES [59], require less computational resources and memory, as they use the same key for encryption and compression. The disadvantage of these algorithms is that a key management scheme is required for key distribution to the communicating parties.

Except the privacy and security issues, energy efficiency is also an important issue in WSNs, as sensors are often battery-operated, and they can be placed in harsh environments (e.g. [60]) where human intervention is not possible. As shown in the literature [61, 62], most of the energy spent in a sensor is due to its wireless transceiver operation. Even if the sensor is not active in a wireless communication, it still needs to receive and decode, up to a certain point, all network packets emitted by its neighboring sensors. Usually, data compression is used so as to minimise the required information for transmission, and to save energy. If security and privacy is required, often encryption follows after compression takes place. At the receiver, decryption and decompression are used, as distinct operations, to recover the initial information.

The last few years Compressive Sensing (CS) has appeared as a new theory that provides encryption and compression in a single step. As shown in [63], if a signal has a sparse representation in one basis, it can be recovered from a small number of projections in a second basis that is incoherent with the first.

Assume that $x \in \mathbb{R}^N$ refers to information collected by a sensor. Suppose that there is a basis Ψ of $N \times 1$ vectors $\{\psi_{i=1}^N\}$ such that $x = \Psi b$, where $b \in \mathbb{R}^N$ is a sparse vector with S non-zero components ($\|b\|_0 = S$). According to CS theory, the information contained in x can be projected using matrix $\Phi \in \mathbb{R}^{M \times N}$, giving $y = \Phi x$, where $y \in \mathbb{R}^M$ is the compressed version of x. As $M \ll N$, the choice of M controls the compression rate of the original data. Furthermore, the compression rate affects the performance of CS, in terms of the reconstruction error. According to Candes et al. [63], an S-sparse signal x can be reconstructed exactly with high probability if $M \geq CS\log(N/S)$, where $C \in R^+$. In any other case, there is a trade-off between the compression rate and the reconstruction error, so, in general, the higher the compression rate is, the higher this error becomes. In general, the compression/encryption using the CS principles is expressed as follows:

$$y = \Phi x = \Phi\Psi b = \Theta b \tag{1.1}$$

where $\Theta = \Phi\Psi$. The original vector b, and consequently the sparse signal x are estimated using the following ℓ_1 norm convex relaxation problem:

$$\hat{b} = \arg\min \|b\|_1 \quad s.t. \quad y = \Theta b. \tag{1.2}$$

Observe that the above problem is an under-determined problem with less equations than unknowns as $M \ll N$.

1.6.1 Computational secrecy of CS

Consider a scenario where a wireless sensor repeatedly collects sensitive data x, and then by using Φ encrypts these data into ciphertext y. Also assume that an attacker is present that passively monitors the wireless channel; thus, being able to capture the encrypted data y transmitted by the legitimate sensor. The goal of the attacker is to guess Φ by examining the transmitted blocks of y. This attack is usually referred as *known ciphertext attack*. The attacker may try to guess Φ by forcing a brute force attack based on the ciphertexts it has collected, and searching over the values for Φ based on a step size. Then, and for each ciphertext, it creates (guesses) a matrix $\Phi^{'}$, and it reconstructs (decrypts) y to $\hat{x} = \Psi\hat{b}$ using (1.2). At this point, the attacker can estimate, through the reconstruction process (see [37] for details), the residual error that can be used as a metric of the reconstruction accuracy. If the residual error is larger than a threshold, it retries the same procedure, otherwise, it stops and assumes that it has guessed the correct matrix Φ. Nevertheless, as shown in [16], for this brute force attack to become feasible, the computation cost is in the order of $O(N^{1.2})$, something that makes this process too expensive.

Another type of attack against a CS crypto-system is an attack based on the symmetry and sparsity structure of matrix Φ (described in [16]). This attack is composed of two phases: During the first phase, the attacker tries to estimate the t leading columns of matrix Φ, assuming that x has t non-zero leading coefficients, and the corresponding coefficients in x such that $\Phi_t x_t = y$. A random permutation of the columns of Φ, and of the corresponding positions in x, produces the same values of y. For this reason, during the second phase, the attacker has to determine the appropriate permutation, so as to find a suitable solution for the over-determined system shown in (1.2). This system has become over-determined as $t < N$. The number of possible permutations requires $C(N,t) \times t!$ possible arrangements that make this attack highly complex.

1.6.2 Information theoretic secrecy of CS

Information theoretic secrecy is based on the statistical properties of a crypto-system providing security even if an attacker has an unbounded processing power. Shannon [64] introduced the idea of perfect secrecy, defining that a crypto-system achieves perfect secrecy if the probability of a plaintext conditioned on the ciphertext, is equal to the *à* priori probability of the plaintext, $P(X = x \mid Y = y) = P(X = x)$. Using the mutual information I, this can be expressed as $I(X : Y) = 0$. The mutual information is used to measure both the linear and non-linear correlation. This is usually difficult to measure but it is a natural measure of the dependence between random variables, considering the whole dependence structure of these variables [65]. Mutual information is computed as follows:

$$I(X : Y) = \sum_{x \in X} \sum_{y \in Y} p_{xy}(x,y) log_2 \frac{p_{xy}(x,y)}{p_x(x) p_y(y)}, \tag{1.3}$$

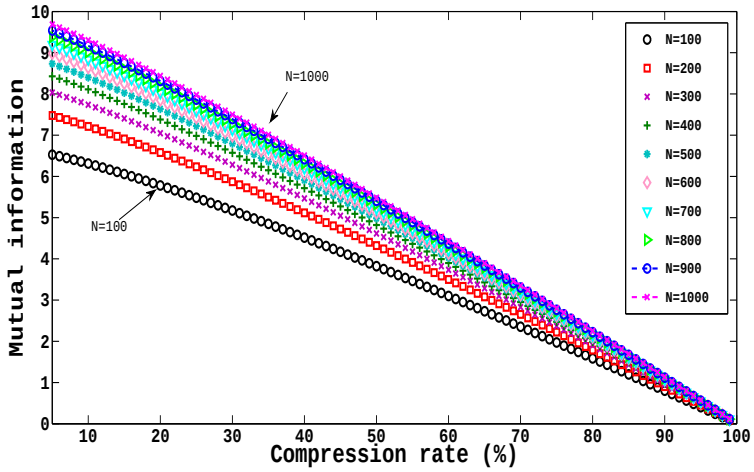

Figure 1.3: Mutual information for an increasing sample length and compression rate

where p_{xy} and p_x denote the joint probability density function (PDF) and marginal PDF, respectively.

As (1.1) shows, ciphertext y is a linear projection of plaintext x so, it is expected that no perfect secrecy can be achieved using CS for encryption. For demonstration purposes, we empirically compute the mutual information of a plaintext and its corresponding ciphertext when applying CS, for different plaintext lengths, and for various compression rates. The compression rate is defined as $\frac{N-M}{M} \times 100$, where N and M are the lengths of the plaintext, and the ciphertext, respectively. Observe in Figure 1.3 that as the compression rate increases, mutual information decreases; hence, higher information secrecy is achieved. This is because when the compression rate increases, ciphertext's length becomes smaller, and linear projections are fewer, so the information leakage from the plaintext to the ciphertext reduces. Furthermore, observe that as the length of the plaintext increases, mutual information increases, because for the same compression rate, more information leakage takes place due to the linear projections.

One would assume that by compressing a plaintext using a higher compression rate, would be the ideal solution in a CS crypto-system. Nevertheless, CS performance, in terms of the reconstruction error, deteriorates as compression increases. The performance is also affected by the sparsity of the plaintext. Figure 1.4 shows the trade-off between the mutual information and the reconstruction error e of CS, defined as $e = \frac{||x-\hat{x}||_2}{||x||_2}$, where x and $\hat{x}$ are the original and reconstructed plaintexts, respectively. Compressing a plaintext using a higher compressive rate can provide higher information secrecy, however, the reconstruction error increases.

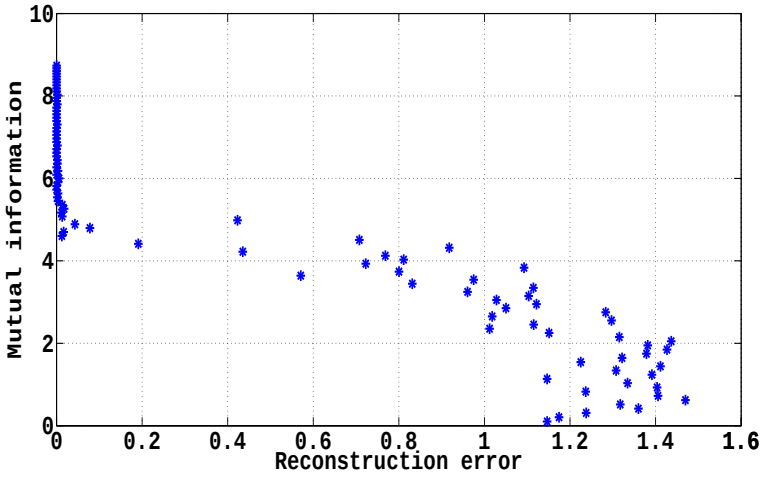

Figure 1.4: Trade-off between the mutual information and the reconstruction error

1.7 Digital Signatures

In this section we will give a quick introduction to the key goals of integrity and authenticity protection on the message level using digital signatures. *Integrity* is violated whenever data or a system is modified in an unauthorised way without being detected. The protection og the integrity of a message against random errors can be fulfilled by a simple checksum like CRC [1]. However, in the presence of an attacker a simple checksum is not sufficient, as an attacker would recalculate the checksum for the maliciously modified message. The mechanism of choice is a so-called protected checksum, that prevents the attacker from being able to recompute the checksum to match malicious changes made [66]. The two most common forms to accomplish this are: keyed hashes, also known as Message Authentication Codes (MAC)[2] and digital signatures. Both can be used to achieve integrity protection against random and malicious errors. To additionally gain a verifiability of origin and also induce accountability for the messages digital signatures are required, as their base is an asymmetric key pair. Accountability on the level of individual devices is important, consider for example that the home owner and the insurance company might want to know who send the message that instructed the roof window to open during the heavy rain and caused the living room to be flooded. When the actuator on the window retains a log of the last signed commands the signature key on the open command can be used to identify the sender.

[1] see also ISO 3309

[2] HMAC is a prominent example see RFC 6151

In this section we start with briefly defining the overall security goals and then go into the details of the challenges and suggested solutions in the light of the IoT. We conclude with a brief summary.

1.7.1 Goals of Integrity Protection and Origin Authentication

The notion of data integrity can be defined as follows:

> ***Data Integrity: The property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner.*** [66].

As a first line of defence it is important to maintain this state. To run systems as complex as the IoT preventive measures are always good. However, attacks or errors can occur on any involved system or communication, e.g. during the transmission and during storage. So it is always needed to have a second line of defence to identify if the state of Integrity has been violated. Hence, the protection mechanisms can be further distinguished into preventive and detective measures. The preventive view of Integrity protection (found for example in classic definitions from *Clark* and *Wilson* [67]) requires mechanisms that prohibit unauthorised modifications upfront. The latter fulfils its duty as it "detects the violation of internal consistency" [68].

In the remainder we will concentrate on digital signatures which are a detective measure. Note, that for the correct and expected behaviour of the IT system the reason for an unwanted modification does not matter. Altered data must become reliably distinguished from unchanged data. Note, that the Integrity notion and thus also the protection mechanisms for Integrity are not concerned with the quality of the data. If information is incorrectly captured into the system data, integrity protection will prohibit undetected modifications to this false data. This limitation of the protection scope of integrity is important to note from an IoT application's perspective. This situation is covered by the notion of *Veracity* [69] introduced by *Gollmann*:

> ***Veracity is the property that an assertion truthfully reflects the aspect it makes a statement about.*** [69]

For an IoT example take the case that someone secured all the sensor readings of the smart device with an integrity protection mechanism. Now, errors or attacks that tamper the readings are detected. However, what is not covered is if the sensed value was actually correct. The sensor could, due to a physical error report wrong values. This data represents false information about the physical entity, hence it does not offer Veracity. However,if no tampering has happened after the wrong value was integrity protected, such the data remains not changed, it still has Integrity.

A digital signature scheme (DSS) is based on asymmetric cryptography. Like in asymmetric encryption, two — related — keys are involved: the secret signature creation key and a related public verification key. The important algorithms of DSS implement the two functionalities: Sign and Verify. The "sign" algorithm takes the secret signing key and the message and generates a signature value. The "verify" algorithm takes the signature value, the message and the public verification key to obtain the result. If the result is positive this means that the signature is valid. A

secure DSS is correct, meaning that for any correctly signed data the signature over the unchanged data must verify as valid. In turn this yields two things: First, the data has not been altered according to an integrity policy and second, the signature value has been created involving the secret key corresponding to public key used for verification. To hold the DSS needs to offer unforgeability; EUF-CMA [70] is an adequate model for this.

If a trusted link between an entity (a human user, a device or also a service[3]) and the public verification key exists, a valid signature on a message implies that the message originated at the entity. This gives origin authentication and can be used to build entity authentication protocols [71]. The reason why other methods, not based on asymmetric keys, miss authenticity of origin is that in a symmetric key based integrity check the verifier needs to know the secret that was used to generate the integrity check value.

The most widely known algorithms are ECDSA (based on elliptic curves) and DSA (based on RSA). Hence, in theory the verifier, as well as the "signer", can generate a valid keyed hash for any message.

1.7.2 Technical Challenges and Solutions for Integrity and Origin Authentication in the IoT

Without claiming completeness, we would like to highlight the following challenges, not ordered in any specific way, and give a hint on how they could be solved technically:

- Challenge 1: constrained devices.
- Challenge 2: loosely coupled system (data gets stored in databases, message queues, and the next step is done later).
- Challenge 3: heterogenous interconnected devices need to work smoothly together.
- Challenge 4: setting up keys and trust relationships.
- Challenge 5: privacy.

Note, there might be still more problems, and also the technical solutions proposed here are just meant as a starting point and no complete survey of all existing and appropriate techniques.

1.7.2.1 Solution 1: constrained devices require efficient implementations of efficient cryptographic primitives like ECC

Overall, Elliptic curve based cryptography (ECC) currently presents itself to be usable in software implementations, and is well supported in today's recent IoT hardware platforms [29, 30, 31]. ECC was presented already in 1986 independently by

[3] "system entity - An active part of a system — a person, a set of persons (e.g., some kind of organization), an automated process, or a set of processes — that has a specific set of capabilities." [66]

Miller [19] and *Koblitz* [20]. NIST has standardised the Elliptic Curve Digital Signature Algorithm (ECDSA) in 2000 [72]. Implementation for wireless sensor networks exist, e.g. TinyECC [21], NanoECC [22] or NIST's ECClight [23] and have been further optimised [24]. As another example of how ECC is the perfect candidate for constrained devices, besides *Ayuso* et al.'s work [24], take the 160bit curve implementation by *Kern* and *Feldhofer* on an RFID-type like devices [25] or the very lightweight ECC based construction for authentication of *Mayer* and *Hess* [73]. NIST's curves have been criticised [27] and the proposal by *Bernstein*, which is additionally claimed to be less problematic for implementers, named Curve25519 with the signature algorithm named Ed25519 is currently en route to standardisation [28].

In this light, we assume ECC signature generation and verification to be a good choice.

1.7.2.2 Solution 2: loosely coupled system requires message-level integrity

IoT sensory information gathered by the devices and the data is forwarded to other devices or to the backbone servers. In a loosely coupled system the data is not immediately processed, but often it is stored in message queues to be picked up later by applications to achieve the desired functionality. For example assume a device with the thermistor to continuously push his readings into a message queue on a cloud server. Asynchronously this message queue is read by several different applications.

The protection of the integrity for those type of loosely connected systems can be achieved by message-level protection mechanisms. Using cryptographically secure digital signature schemes allows verifying that data was not modified in unauthorised ways. Additionally, you can use a DSS to gain origin-authentication knowing which entity, hence which device, signed the data. Of course, if for privacy reasons this fine-grained identifiability is not wanted or permitted, than other cryptographic primitives are needed, e.g. like group-signatures [74].

1.7.2.3 Solution 3: heterogenous interconnected devices need to work smoothly together requires standardising formats and algorithms

The goal must be to apply integrity protection to the data at the earliest point, i.e., at the device [6], in an end-to-end fashion, all the way downstream to the latest point possibly in the IoT data processing chain. That means that some intermediary devices/nodes might not understand the integrity protection as they see no need to check it. As Figure 1.2 depicts, the MiddleWare or the GateWay might, but need not be the ones who verify the integrity. Hence, this requires that adding security by adding the digital signature to the data, shall not break existing workflows. As from Challenge 2 we already know that the protection of integrity must be on the message level, e.g. each temperature reading carries a signature. So non-signature aware processing steps for the data in the IoT value chain shall still be able access the data that contains the signature in the same manner as if it would not be signed.

Assuming that the currently wide spread use of JSON as a transport format, e.g. for COAP, [75] one solution is to add the digital signature as (one or more) additional element(s) into the JSON object that the sensor emits [76]. How the signature gets encoded must be standardised. We would like to point to existing work done in the non-constrained world, e.g. JSON Web Signature (JWS) [77] that is currently discussed in IETF's JOSE working group as an draft. Also you might want to look at CBOR aimed for small code size and small message size described in RFC 7049 [78]. This is used by the IETF's working group COSE [79] to slim down JOSE. To the best of the authors knowledge there is still no agreed solution and future work to do regarding initial ideas[4], and then the still open standardisation of a suitable format for the IoT domain. Also for each curve algorithm[5] the keys must be represented in an interchangeable format. ECC can be put in X.509 certificates, see for example [81].

1.7.2.4 Solution 4: setting up keys and trust relationships

As mentioned before, digital signature schemes require asymmetric keys; in other words a pair of public verification key and secret signing key for each device. This requires to generate and manage even more keys than we do today for users and servers. This is a problem that shall not be underestimated, but potentially thats a technical distribution problem that can be managed, e.g. encoding keys into optically scannable QR-codes or transmittable via RFID.

More problematic than the distribution of public keys is to solve the problem of gaining trust into such a public key. A party that verifies a signature using a certain public key shall be convinced that they can deduct trust in a public key. Wether this is done by a hierarchically organised Certificate Issuer, e.g. the CAs in the PKI of the Internet today, or more decentralised by a web-of-trust, e.g. like with PGP, is not important for a secure operation of digital signatures in the IoT. But it must be solved, as without a way to establish trust into the public key used for integrity verification the protection falls to man-in-the-middle attacks. An attacker shall not be able to convince IoT devices that his public key — for which the attacker has the secret signature generation key and thus can produce valid signatures — is trusted to sign firmware updates, or sensitive commands.

As such, the IoT must first bring the trusted keys of central IoT system components, like a firmware update service, onto each and every IoT device. Then they need to be stored securely such that they can not be exchanged by an attack. The latter requires a secure tamperproof storage solution for the device.

1.7.2.5 Solution 5: Privacy for the signing entity by group signature; privacy for data by malleable signatures

The problem of privacy in the IoT does not stop to be important when considering the implementation and integration of integrity protection. This is two-fold: On the one hand, the device that generated the signature, by means of a different asymmetric signature generation key unique for each device, may be needed to stay pseudony-

[4] an initial not yet length-optimized format was presented in [76]

[5] a list of algorithm descriptors is already defined in [80]

mous while still being able to check that a certain trusted group of devices had signed it. This can be achieved by a choice of the right cryptographic primitives, e.g. using group signatures. On the other hand, the data that is signed may be subject to changes for the name of privacy but this shall not completely destroy the integrity of the message. This can be achieved using malleable signatures [82]. They allow the removal of specified parts by a specified mandated party (like a delegation) of a signed message without removing the integrity protection. There are manyfold cryptographic schemes, for a comparison of two major strands of those see [83].

1.7.3 Summary: End-to-End Integrity and Authenticity based on ECC

Achieving Integrity for the IoT has two core steps towards its use: (1) adequately lightweight cryptographic primitives to run on the constrained devices, (2) finding and then standardising efficient signature transport formats and algorithms. Elliptic curve based crypto (ECC) currently is the promising and already well supported (in hardware accelerators and software library). Regarding step two many groups discussing this currently, so there will likely be standards to adhere to in the near future. Furthermore, it has adjacent steps that need to be climbed as well: (3) putting trust into public keys and (4) their initial and ongoing distribution. The question of secured storage of keys is the IoT related part and a question regarding the IoT's hardware capabilities, however to induce trust it needs organisational methods, like the secure operation of a CA. Those organisational methods of course can be legally enforced as they can only be technically supported.

The RERUM project has created initial prototypes of an integration of ECC based digital signatures into the IoT ecosystem. Initial results obtained based on ECC digital signature algorithms like NIST's secp256r curve and ed25519 look promising[6].

1.8 Datagram Transport Layer Security - DTLS

Securing communication is a far from a trivial task. Difficulties may arise from a wide range of technical or so-called human-factor aspects, including (but not limited to) the design and selection of applicable cryptographic primitives, and the use carefully investigated cryptographic protocols and applications. Moreover, the possible designs used to secure communications should not only be investigated in theory, their practical implementations should also be considered. In such a case, a designer should additionally take into consideration, e.g., the efficiency of an algorithm on a vast rage of different platforms and (especially for embedded devices) their protection against implementation attacks [84].

The level of complexity associated with achieving secure communication is already high for an ordinary workstation or a server, and becomes even more challenging in IoT systems. This is mostly due to the fact that new design criteria, such as

[6]currently the measurements within RERUM are still running but see [76] for some rough runtime estimations

the power consumption of a device or its computational power, have to be taken into consideration. In many cases these additional limitations are simply too restrictive to run cryptographic primitives and protocols efficiently.

1.8.1 DTLS Protocol for Internet of Things

Probably one of the best-known and widely deployed cryptographic protocol, which shapes secure communication over the Internet is the Transport Layer Security (TLS) protocol [85]. In general, it allows two parties (called 'peers' or a client and a server) to establish and carry out a secure communication. The TLS protocol is widely used by applications such as Internet browsers, as well as more stream-oriented applications such as voice-over-IP services.

TLS belongs to a group of TCP/IP protocols and (considering the standard OSI model) operates in the session and presentation layers. In fact, within TLS, two protocols named the TLS Record Protocol and TLS Handshake Protocol can be distinguished. The record protocol resides in the fore-mentioned presentation layer and provides secure and reliable connection for the application layer. The handshake protocol, however, resides in the session layer and is mostly responsible for establishing the connection, e.g., for parameter negotiation and for the authentication of the peers. Since TLS runs over a (reliable) TCP connection, no extra measure on ordering and loosing a packet inside TLS itlsef is needed.

In general, whenever a high-level protocol uses TCP as an underlying transport protocol, the correct packet order is maintained and no single packet is supposed to be lost during the transmission. This TCP feature is especially important for many cryptographic protocols, where a single packet loss or a wrong packet order might have impact on the overall message correctness. Unfortunately, the TCP reliability property increases the overheads of the protocol stack, thus in some scenarios more lightweight protocols such as the User Datagram Protocol (UDP) could be used instead. Since UDP features smaller packet headers, requires less memory and has more compact code footprint, it is better suitable for constrained devices which are ubiquitous in IoT systems.

Regrettably, the TLS protocol itself is not applicable (without modification) to UDP connections. To address this issue a new protocol called DTLS [86] was designed. The new protocol mirrors most of the futures and design choices of TLS and extends them with a mechanism to allow packet retransmissions and reordering. This allows the DTLS protocol to run over UDP, providing reliable connection to high-level protocols. Similarly to TLS, DTLS provides authenticity, integrity and confidentiality to the application layer protocols, e.g., it might be used as the underlying protocol to the Constrained Application Protocol (CoAP) [75]. The CoAP is designed especially for resource-constrained environments.

Due to its features, the DTLS protocol for resource-constrained devices has recently gained attention and it is currently investigated by the standardization group 'DTLS In Constrained Environments' (DICE), which is one of the community groups within the Internet Engineering Task Force (IETF). The DICE group aims to specify a DTLS protocol profile suitable for IoT systems [87].

Reassembling TLS, DTLS consists of the handshake and record protocols. The former is used to negotiate a cipher suite, exchange keys and authenticate the peers, whereas the latter is used to protect traffic data. The handshake protocol uses series of messages (called flights), which are exchanged between communication parties in order to achieve above-mentioned goals. To accomplish these goals, the protocol can select and use public- and private-key schemes. The selection of a suitable mode and credential types has a big impact on the overall handshake performance, thus it is crucial to find an appropriate profile for IoT systems. There are three possible credential types defined by DTLS: a private-key credential type called pre-shared secret, and two public-key credential types called raw public keys and X.509 certificates.

In a pre-shared secret scenario, secret keys and identifiers, which are needed to establish a secure communication between two peers, are deployed and stored in devices (using a trusted channel) before communication. Considering resource-constrained devices, which are likely to be used in IoT, this approach could be, in general, be more efficient – in terms of computation power needed by a device – than a public-key approach. One may also find this solution well-suited to strictly controlled deployment scenarios, where pre-defined connections might make general key management over-engineered. However, in the large scale IoT deployments a pre-shared key solution might not scale enough, and additionally, the large number of keys needing storing might not be suitable for the constrained memory available in such devices.

In contrast to a pre-shared key, in a raw public key scenario, a client and a server use a public/private key pair, which might be generated and further installed on a device by a device manufacturer. To a certain extent this could be seen as a more lightweight version of the X.509 certificates approach, trading off the flexibility provided by X.509 certificates for a better performance. In general, a raw public key credential type might use out-of-band distribution and only a small subset of the X.509 certificate structure. This positively impacts performance and storage requirements, which is essential in IoT systems, but on the other hand limits flexibility provided by X.509 certificates.

The X.509 certificate credential type is the other option (on top of the above-mentioned raw key scenario), where peers use public/private key pair to established communication. To use this mode in the DTLS protocol, a public-key infrastructure, including a Certificate Authority (CA), needs to be deployed. Moreover, devices are augmented with a list of so-called trust anchors to validate certificates. The certificates are deployed in the form specified by the X.509 standard which includes all meta-data information needed. This implies a requirement for increased on-device storage as well as the extra time need to parse a certificate and validate with the CA. Although the X.509 type provides a significant level of flexibility w.r.t. a key management, its full deployment is usually very changeling considering the limitation of resource-constrained environments.

1.8.2 Summary

Although DTLS seems as a good candidate protocol for standardization aiming to secure communication in IoT systems, its implementation and deployment remains a challenge, mostly due to a constrained nature of resources used in such systems. On the one hand, DTLS provides the possibility to use private-key mode, which could be well-suited in terms of, i.e., computation power, on the other hand this solution suffers from the well-known problem of key management scalability. The solution for a key management issue might be mitigated by using public-key schemes and supporting infrastructure, but public-key operations are costly in general and are still far from being efficient enough for very constrained devices. Advances in EC-based cryptography, including the new EC-based designs such as Curve25519/Ed25519 [28] or the new set of elliptic curves relevant for cryptographic use [88], might bring more suitable solutions for IoT systems.

1.9 Discussion

Security has become a main research area in the Internet of Things world the last few years. However, the work until now focused only on access control mechanisms for users that need to have access to data that are stored in IoT middlewares, without much concern for the leaf IoT devices. It has been proved though that an IoT system cannot be fully secured without securing the embedded devices themselves. Regardless of how securely the information is sent from the middleware to the applications, the numerous vulnerabilities of the constrained IoT devices leave a whole lot of opportunities to attackers to intercept the data, to hack the devices, to steal personal user information or to affect system decisions. This section presented a high-level framework for increasing the security in the communication layer of IoT systems, with a focus on enhanced lightweight encryption, integrity protection and overall communication security. The benefits of the proposed framework is that it comprises lightweight mechanisms that can be easily applied on constrained IoT devices and can work either as standalone components or as an integrated security system for increased protection against attacks. Various design choices on where and how to implement these mechanisms were also proposed and the final decision is left to the system administrator, considering the applications that his system supports. This framework can be part of an overall IoT cross-layer security framework, contributing mainly to the protection of the data at the earliest point, something that can significantly increase the overall trustworthiness of the system and IoT in general. Only then, the general public and the business stakeholders can have the necessary incentives for adopting and using or investing in this new set of technologies that can drastically improve the everyday life of people.

Bibliography

[1] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[2] Cisco, "The internet of everything. connections counter," *http://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342*, 2013.

[3] H. Report, "Internet of things research study, hp report (2014)," *http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf*, 2014.

[4] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011.

[5] C. M. Medaglia and A. Serbanati, "An overview of privacy and security issues in the internet of things," in *The Internet of Things*. Springer, 2010, pp. 389–395.

[6] H. C. Pöhls, V. Angelakis, S. Suppan, K. Fischer, G. Oikonomou, E. Z. Tragos, R. Diaz Rodriguez, and T. Mouroutis, "Rerum: Building a reliable iot upon privacy-and security-enabled smart objects," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2014 IEEE*. IEEE, 2014, pp. 122–127.

[7] O. Vermesan and P. Friess, *Building the Hyperconnected Society. Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*. River Publishers, 2015.

[8] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. Kranenburg, S. Lange, and S. Meissner, *Enabling things to talk: designing IoT solutions with the IoT architectural reference model*. Springer, 2013.

[9] S. Menoret *et al.*, "Final architecture reference model."

[10] B. consortium, "Integrated system architecture and initial pervasive butler proof of concept."

[11] D. Rotondi *et al.*, "Final framework architecture specification."

[12] P. Dimitropoulos *et al.*, "Openiot detailed architecture and proof-of-concept specifications."

[13] F. Carrez *et al.*, "Conceptual model and reference architecture."

[14] "Xilinx zynq-7000 all programmable soc zc702."

[15] A. Skarmeta *et al.*, "Initial architecture specification."

[16] A. Orsdemir, H. Altun, G. Sharma, and M. Bocko, "On the security and robustness of encryption via compressed sensing," in *Proc. of MILCOM*, 2008, pp. 1–7.

[17] Y. Rachlin and D. Baron, "The secrecy of compressed sensing measurements," in *Proc. of Allerton Conf. on Communication, Control, and Computing*, 2008, pp. 813–817.

[18] V. Cambareri, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Low-complexity multiclass encryption by compressed sensing," *IEEE Transactions on Signal Processing*, vol. 63, pp. 2183–2195, 2015.

[19] V. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology-CRYPTO85 Proceedings*. Springer, 1986, pp. 417–426.

[20] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[21] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*. IEEE, 2008, pp. 245–256.

[22] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks," in *Wireless sensor networks*. Springer, 2008, pp. 305–320.

[23] National Institute of Standards and Technology (NIST), "ecc-light-certificate Library," https://github.com/nist-emntg/ecc-light-certificate, 2014.

[24] J. Ayuso, L. Marin, A. Jara, and A. F. G. Skarmeta, "Optimization of public key cryptography (rsa and ecc) for 16-bits devices based on 6lowpan," in *1st International Workshop on the Security of the Internet of Things, Tokyo, Japan*, 2010.

[25] T. Kern and M. Feldhofer, "Low-resource ECDSA implementation for passive RFID tags," in *17th IEEE International Conference on Electronics, Circuits, and Systems (ICECS'10)*. IEEE, 2010, pp. 1236–1239.

[26] M. Braun, E. Hess, and B. Meyer, "Using elliptic curves on RFID tags," *International Journal of Computer Science and Network Security*, vol. 2, pp. 1–9, 2008.

[27] D. J. Bernstein, T. Chou, C. Chuengsatiansup, A. Hülsing, T. Lange, R. Niederhagen, and C. van Vredendaal, "How to manipulate curve standards: a white paper for the black hat," Cryptology ePrint Archive, Report 2014/571, Tech. Rep., 2014.

[28] N. Moeller and S. Josefsson, "IETF Draft: EdDSA and Ed25519," https://tools.ietf.org/html/draft-josefsson-eddsa-ed25519-02, Feb. 2015.

[29] Atmel Corporation, "ECC-based Devices," http://www.atmel.com/products/security-ics/cryptoauthentication/ecc-256.aspx, 2015.

[30] Openmote.org, "Openmote CC2538," http://www.openmote.com/hardware/openmote-cc2538-en.html, 2014.

[31] Zolertia, "Zolertia ReMote," http://zolertia.io/product/hardware/re-mote, 2015.

[32] E. Z. Tragos, V. Angelakis, A. Fragkiadakis, D. Gundlegard, C.-S. Nechifor, G. Oikonomou, H. C. Pöhls, and A. Gavras, "Enabling reliable and secure iot-based smart city applications," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2014 IEEE International Conference on*. IEEE, 2014, pp. 111–116.

[33] E. J. Candè and M. B. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21–30, 2008.

[34] D. E. Fu and J. A. Solinas, "Ike and ikev2 authentication using the elliptic curve digital signature algorithm (ecdsa)," 2007.

[35] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology–CRYPTO 2004*. Springer, 2004, pp. 41–55.

[36] H. C. Pöhls, S. Peters, K. Samelin, J. Posegga, and H. de Meer, "Malleable signatures for resource constrained platforms," in *Information Security Theory and Practice. Security of Mobile and Cyber-Physical Systems*. Springer, 2013, pp. 18–33.

[37] P. Charalampidis, A. G. Fragkiadakis, and E. Z. Tragos, "Rate-adaptive compressive sensing for iot applications," in *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*. IEEE, 2015, pp. 1–5.

[38] V. Kumar, G. Oikonomou, T. Tryfonas, D. Page, and I. Phillips, "Digital investigations for ipv6-based wireless sensor networks," *Digital Investigation*, vol. 11, Supplement 2, no. 0, pp. S66–S75, August 2014, fourteenth Annual DFRWS Conference.

[39] G. Montenegro, N. Kushalnagar, J. W. Hui, and D. E. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," RFC 4944, Sep. 2007.

[40] J. Hui (Ed.) and P. Thubert, "Compression format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," RFC 6282, Sep. 2011.

[41] Z. Shelby (Ed.), S. Chakrabarti, E. Nordmark, and C. Bormann, "Neighbor discovery optimization for low-power and lossy networks," RFC 6775, November 2012.

[42] T. Winter (Ed.), P. Thubert (Ed.), A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks," RFC 6550, Mar. 2012.

[43] P. Nikander (Ed.), J. Kempf, and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats," RFC 3756, May 2004.

[44] A. Alsa'deh and C. Meinel, "Secure neighbor discovery: Review, challenges, perspectives, and recommendations," *IEEE Security Privacy*, vol. 10, no. 4, pp. 26–34, July 2012.

[45] A. Le, J. Loo, A. Lasebae, M. Aiash, and Y. Luo, "6LoWPAN: a study on QoS security threats and countermeasures using intrusion detection system approach," *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1189–1212, Sep. 2012.

[46] A. Dvir, T. Holczer, and L. Buttyan, "VeRA - version number and rank authentication in RPL," in *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, October 2011, pp. 709–714.

[47] T. Tsao, R. K. Alexander, M. Dohler, V. Daza, A. Lozano, and M. Richardson (ed), "A security threat analysis for routing protocol for low-power and lossy networks (RPL)," Internet Draft (version 06), December 2013, (draft-ietf-roll-security-threats).

[48] L. Wallgren, S. Raza, and T. Voigt, "Routing Attacks and Countermeasures in the RPL-Based Internet of Things," *International Journal of Distributed Sensor Networks*, vol. 13, no. 794326, 2013.

[49] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time Intrusion Detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, November 2013.

[50] T. Clausen and U. Herberg, "Some Considerations on Routing in Particular and Lossy Environments," in *Proc. 1st Interconnecting Smart Objects with the Internet Workshop*, Mar. 2011.

[51] P. Ilia, G. Oikonomou, and T. Tryfonas, "Cryptographic key exchange in ipv6-based low power, lossy networks," in *Proc. Workshop in Information Theory and Practice (WISTP 2013)*, ser. Lecture Notes in Computer Science, vol. 7886. Springer, May 2013, pp. 34–49.

[52] T. O'Donovan, J. Brown, F. Büsching, A. Cardoso, J. Cecílio, J. D. Ó, P. Furtado, P. Gil, A. Jugel, W.-B. Pöttner, U. Roedig, J. S. Silva, R. Silva, C. J.

Sreenan, V. Vassiliou, T. Voigt, L. Wolf, and Z. Zinonos, "The ginseng system for wireless monitoring and control: Design and deployment experiences," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 4:1–4:40, December 2013.

[53] E. Byres and J. Lowe, "The myths and facts behind cyber security risks for industrial control systems," in *Proc. of the VDE Kongress*, vol. 116, 2004.

[54] B. Triki, S. Rekhis, and N. Boudriga, "Digital investigation of wormhole attacks in wireless sensor networks," in *Proc. Eighth IEEE International Symposium on Network Computing and Applications (NCA 2009)*, 2009, pp. 179–186.

[55] F. Mouton and H. Venter, "A prototype for achieving digital forensic readiness on wireless sensor networks," in *Proc. AFRICON, 2011*, 2011, pp. 1–6.

[56] S. Rekhis and N. Boudriga, "Pattern-based digital investigation of x-hole attacks in wireless adhoc and sensor networks," in *Proc. International Conference on Ultra Modern Telecommunications & Workshops (ICUMT '09)*, 2009, pp. 1–8.

[57] S. Dawans and L. Deru, "Demo Abstract : Foren6, a RPL/6LoWPAN Diagnosis Tool," in *Proc. 11th European Conference on Wireless Sensor Networks (EWSN)*, Feburary 2014.

[58] B. Kaliski, "The mathematics of the rsa public-key cryptosystem," *RSA Laboratories*.

[59] K. Raeburn, "Advanced encryption standard (aes) encryption for kerberos 5," *RFC 3962*, 2005.

[60] G. Werner, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing, Special Issue on Data-Driven Applications in Sensor Networks*, vol. 10, pp. 18–25, 2006.

[61] A. Dunkels, "The contikimac radio duty cycling protocol," Swedish Institute of Computer Science, Technical Report, 2011.

[62] A. Fragkiadakis, I. Askoxylakis, and E. Tragos, "Secure and energy-efficient life-logging in wireless pervasive environments," in *Proc. of the 1st International Conference on Human Aspects of Information Security, Privacy and Trust*, 2013, pp. 306–315.

[63] E. Candes and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[64] C. Shannon, "Communication theory of secrecy systems," *Bell System Technical Jounal*, vol. 28, pp. 656–715, 1949.

[65] J. Williams and Y. Li, *Estimation of Mutual Information: A Survey*. Rough Sets and Knowledge Technology, Lecture Notes in Computer Science, Springer, 2009.

[66] R. Shirey, "RFC 4949–Internet Security Glossary," 2007.

[67] D. D. Clark and D. R. Wilson, "A comparison of commercial and military computer security policies," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 184, 1987.

[68] E. Michiels, "ISO/IEC 10181-6: 1996 Information technology — Open Systems Interconnection — Security frameworks for open systems: Integrity framework," *ISO Geneve, Switzerland*, 1996.

[69] D. Gollmann, "Veracity, plausibility, and reputation," *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems*, pp. 20–28, 2012.

[70] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal on Computing*, vol. 17, pp. 281–308, 1988.

[71] D. Gollmann, "What do we mean by entity authentication?" in *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*. IEEE, 1996, pp. 46–54.

[72] National Institute of Standards and Technology (NIST), "PUB FIPS 186-4. Digital signature standard (DSS)," 2011.

[73] B. Meyer and E. Hess, "United States Patent 8,850,213," http://www.uspto.gov/web/patents/patog/week39/OG/html/1406-5/US08850213-20140930.html, Jul. 2014.

[74] D. Chaum and E. Van Heyst, "Group signatures," in *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*. Springer-Verlag, 1991, pp. 257–265.

[75] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (CoAP)," RFC 7252, June 2014.

[76] H. C. Pöhls, "Json sensor signatures (jss): End-to-end integrity protection from constrained device to iot application," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2015 9th International Conference on*. IEEE, 2015, pp. 306–312.

[77] M. Jones, J. Bradley, and N. Sakimura, "IETF draft: JSON Web Signatures (JWS)," https://tools.ietf.org/html/draft-ietf-jose-json-web-signature-41, Jan. 2015.

[78] C. Bormann and P. Hoffman, "Concise Binary Object Representation (CBOR)," RFC 7049, Internet Engineering Task Force, 2013. [Online]. Available: https://tools.ietf.org/html/rfc7049

[79] C. Bormann, "IETF draft: CBOR Object Signing and Encryption (COSE)," https://tools.ietf.org/html/draft-bormann-jose-cose-00, Oct. 2014.

[80] M. Jones, "IETF draft: JSON Web Algortihms (JWA)," https://tools.ietf.org/html/draft-ietf-jose-json-web-algorithms-40, Jan. 2015.

[81] S. Blake-Wilson, D. Brown, and P. Lambert, "Use of elliptic curve cryptography (ecc) algorithms in cryptographic message syntax (cms)," Tech. Rep., 2002.

[82] H. C. Pöhls and M. Karwe, "Redactable signatures to control the maximum noise for differential privacy in the smart grid," in *Proc. of the 2nd Workshop on Smart Grid Security (SmartGridSec 2014)*, ser. Lecture Notes in Computer Science (LNCS), J. Cuellar, Ed., vol. 8448. Springer International Publishing, 2014.

[83] H. de Meer, H. C. Pöhls, J. Posegga, and K. Samelin, "On the relation between redactable and sanitizable signature schemes," in *ESSoS*, ser. LNCS, vol. 8364. Springer, 2014, pp. 113–130.

[84] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology–CRYPTO99*. Springer, 1999, pp. 388–397.

[85] T. Dierks and E. Rescorla, "The transport layer security (TLS) protocol version 1.2," RFC 5246, August 2008.

[86] N. Modadugu and E. Rescorla, "The design and implementation of datagram TLS," in *NDSS*, 2004.

[87] Internet Engineering Task Force IETF, "TLS/DTLS profiles for the internet of things," https://tools.ietf.org/pdf/draft-ietf-dice-profile-17.pdf, October 2015.

[88] J. W. Bosand, C. Costello, P. Longa, and M. Naehrig, "Selecting elliptic curves for cryptography: An efficiency and security analysis," *Journal of Cryptographic Engineering*, pp. 1–28, 2014.