


One main application of smart home IoT networks is telehealth, which is timely given the current pandemic situation and increasing healthcare costs. To ensure security of IoT smart homes, it has been suggested that silicon-based physically unclonable functions (PUF) be incorporated in the IoT devices themselves. PUFs are used as the main technique for establishing device authentication and secure key exchange as well as any higher level security protocols. This chapter provides an analysis of the characteristics and performance of SRAM physically unclonable functions. The analysis takes into account several factors such as the static or slowly-varying random process variations as well as the dynamic CMOS noise sources. The main parameters affecting the performance are identified and techniques used to measure them at the fabricator and in the field are explained. Three algorithms are proposed for choosing the set of challenges and the corresponding responses. The three algorithms are: Algorithm #1: single challenge; Algorithm #2: repeated challenge; and Algorithm #3: repeated challenge with bit selection. The last algorithm manages to eliminate the bit errors in the response and hence will not require the use of error correction coding often used in secure sketch or fuzzy extractor methods that have previously been proposed. The use of physically unclonable functions, coupled
with the proposed algorithms, provide a layer of protection against the common IoT attacks and the novel deep learning attacks that EW claimed to be a serious security threat to IoT devices in telehealth applications.

3.1. Introduction

An emerging application of smart homes is telehealth, where healthcare delivery is extended to serve stay-at-home patients and remote or isolated communities. Telehealth is motivated by the escalating healthcare costs and the fact that many patients can not afford long-term hospital stays and prefer staying in their homes or within their remote communities. Telehealth relies very heavily on equipping the home with smart IoT devices that can sense the patient’s vitals and can also deliver medication in a secure environment that is immune to cyber attacks. This approach allows us to reach out to many disadvantaged communities, thereby democratizing healthcare, as well as leading to reduced costs and speedy patient recovery times (Ellenbecker et al. 2008; National Institute on Aging 2020).

IoT devices used in smart homes are considered the weakest link in the security protocols implemented. As a result, contemplating the implementation of critical telehealth services in a smart home is very risky due to the device limitations of the Internet of Things (IoT). Some of the limitations include:

1) Limited resources, such as computer processing capabilities, which often prevent the implementation of secure key exchange algorithms that use complex elliptic functions.

2) Storing secret keys in non-volatile memory (NVRAM) is considered a security gap since simple memory attacks can reveal those secret keys. Furthermore, these secret keys are are hard to update since the NVRAM must be reprogrammed.

3) Users often do not customize or update each IoT device password or operating system firmware and rely solely on factory-set defaults. This is what system attackers first look for to launch their attacks.

4) IoT devices are located in unsecured premises and can be subject to theft, counterfeiting and reverse engineering.

These limitations impact the effectiveness of security protocols for both authentication and secure key exchange. A very promising technique for endowing a simple IoT device with a unique identity and the ability to secure secret keys without using NVRAM is to use silicon-based physically unclonable functions (PUF).

There are many types of PUFs based on different physical phenomena such as optical, acoustical and electrical. However only silicon PUFs implemented as electric circuits are practical for inexpensive implementations on simple IoT devices. Silicon PUFs are practical means of adding unique, unclonable identities to IoT devices. This
is equivalent to biometrics in humans, such as iris, retina, voice, facial or fingerprint. PUFs not only help to authenticate IoT devices, but also aid in storing secret keys in the way a PUF is constructed. Traditionally, secret keys are stored in IoT devices using NVRAM. The disadvantage of NVRAM is the ability of an attacker to extract the secret keys, using many techniques such as memory persistence, reverse engineering, etc. A very attractive property of PUFs is their tamper-resistance which provides immunity from reverse engineering attacks that aim to extract the unique device response. The unique response of the PUF prevents the manufacturer, the user and the attacker from duplicating the PUF function, even when the PUF hardware design and structure are known.

Authentication using PUFs is based on establishing a challenge-response pair (CRP) where a set of challenges and their associated unique response is established by the device manufacturer. This dataset is then shared with a trusted certification authority (CA) for later use by administrators of the telehealth system to construct a secure and trusted system.

There are several criteria for CRP establishment:

1) Several CRP must be established so that each CRP is used only once to prevent attackers from forging a valid response by observing past CRP activities.

2) The number of bits for each response must be “large enough” to be able to establish enough Hamming distance (HD) separation between valid devices and counterfeit ones.

3) Techniques must be established to remove the inevitable dynamic noise from the PUF response to be able to match the noisy response to the one provided by the manufacturer and stored at a CA.

4) Algorithms must be provided to extract a high-entropy stable and repeatable secret key from a noisy low-entropy response.

The ability to construct inexpensive PUFs for IoT edge devices allows us to impart a unique device identity (ID), which is used for device authentication and developing stable and secure session keys. A very significant advantage is that the session key is obtained at the beginning of each session without the use of NVRAM. The key will be shared between the device and the authenticator through the use of publicly available helper data that will not compromise either the key or the device response.

Ensuring security of telehealth systems is hard, since many devices are distributed in insecure locations. Many types of attacks become feasible, such as eavesdropping, theft, tampering, man-in-the-middle, denial of service, etc. Central to ensuring security is authentication and key exchange. Cryptographic protocols are based on primitive operations such as block ciphers, stream ciphers and cryptographic hash functions. These primitive operations rely on storing a secret key stored in non-volatile memory, which proves to be their Achilles heel, especially for unsecured IoT devices (Delvaux 2017b).
The use of PUFs for mutual authentication in IoT devices has been the recognized solution to endowing IoT devices with a unique identity, akin to a fingerprint or retina image for human users. A PUF serves to authenticate a device and also provides a measure of tamper resistance (Gassend et al. 2002; Ravikanth et al. 2002; Guajardo et al. 2007; Suh and Devadas 2007; Maes et al. 2009, 2012; Maes 2013; Herder et al. 2014; Delvaux 2017b). Operation of the PUF relies on a challenge-response pair (CRP), where the server issues a challenge and the IoT device, or client, provides a response that is unique to the device. The problem with PUF response is it is noisy but has low entropy. Therefore, techniques have been developed to recover reliable and stable response from the noisy response using fuzzy extractors or secure sketch (Linnartz and Tuyls 2003; Boyen 2004; Dodis et al. 2004, 2008). The advantage of the fuzzy extractor is that it also serves to generate a secret key with high entropy from the low-entropy noisy response.

Contributions: The contributions of this chapter can be summarized as follows:

1) Novel statistical modeling and analysis of SRAM PUFs is presented. The model includes the effects of static random process variations and dynamic CMOS noise.

2) The main physical, device and system parameters affecting the PUF response are identified and techniques to estimate them are presented for both the IoT device manufacturer and for the IoT device user in the field.

3) A novel NOR-based SRAM PUF cell design is proposed that enables rapid device resetting at a speed matching the operating speed of the system and does not require the waste of too much delay or energy resetting the entire SRAM.

4) Three algorithms are proposed for generating the challenge response pairs. The techniques illustrate the impact of system parameters in uniquely identifying valid devices from counterfeit ones.

5) A discussion is provided on how to harden SRAM PUF against typical IoT attacks and deep learning attacks in particular.

Organization: The rest of this chapter is structured as follows. In section 3.2 we discuss the literature related to the use of PUFs for authentication and secure key exchange. In section 3.3 we review the architecture of a telehealth system where the smart home is the target for the healthcare delivery. In section 3.4 we discuss physically unclonable functions and using secure sketch and fuzzy extractors to remove the dynamic noise from the PUF response. In section 3.5 we discuss the use of convolutional coding as a means of generating the helper data without revealing the IoT device response when a challenge is issued. In section 3.6 the structure of SRAM PUFs is presented and a novel NOR-based SRAM is discussed. A statistical model of the SRAM PUF is also developed. In section 3.7 we propose three algorithms for issuing the PUF challenge-response pair (CRP) data and their effect on system design. In section 3.8 we discuss the attacks targeting smart homes, especially deep learning attacks.
3.2. Related literature

A high-level authentication and key exchange protocol for a smart home IoT system was recently proposed by Fakroon et al. (2020, 2021). The protocol used a two-factor authentication scheme that preserved user anonymity and untraceability. In the 2020 publication, the IoT edge devices were assumed to have secret keys stored in NVRAM. On the other hand, the 2021 publication assumed the secret keys could be derived from a built-in PUF that gave the IoT edge devices unique IDs. Security analysis of the scheme was conducted through formal analysis using the Burrows-Abadi-Needham logic (BAN), informal analysis and model check using the automated validation of Internet security protocols and applications (AVISPA) tool. A review of PUF-based security techniques can be found in Dodis et al. (2004, 2008). The authors discussed how to use a low-entropy PUF response to generate secure keys with high entropy. Secret key extraction techniques used fuzzy extractors to obtain session keys from the noisy PUF responses.

PUFs are classified as strong PUFs and weak PUFs as explained by Delvaux et al. (2014) and Delvaux (2017b). The discussion discussed the impact of strong and weak PUFs on device authentication and secure key exchange. A discussion was also provided about the helper data algorithm and how it can be used to obtain high-entropy stable keys from noisy, low-entropy PUF responses. At a different level, the abstraction of PUF operation as a one-way functions can be found in Ravikanth (2001) and Ravikanth et al. (2002). They compared algorithmic one-way functions (e.g. RSA encryption) with physical one-way functions (e.g. PUFs). A discussion of silicon PUFs can be found in Gassend et al. (2002). The discussion focused mainly on delay-based PUFs such as arbiter PUF and ring oscillator PUF. The authors also discussed helper data, which is used to generate session keys from PUF responses.

An initial attempt at analyzing delay-based PUFs can be found in Suh and Devadas (Suh and Devadas 2007). The analysis considered the need to use each CRP only once. Techniques were proposed to generate a sufficiently large number of responses through increasing the number of options to configure the circuit delays. The authors also discussed low-cost authentication techniques that do not require the use of the more expensive cryptographic primitives.

It is interesting to explore how PUFs can be incorporated using the popular FPGA technology. This is especially true for SRAM PUFs, which could make use of the built-in block RAM provided in many FPGA modules. However, resetting the SRAM might not be a simple matter, since this was not part of the design requirements of an FPGA block RAM (Xilinx 2021). Guajardo et al. (2007) studied SRAM PUF structures implemented in FPGA technology. To overcome noise associated with the
response of the PUF a fuzzy extractor was used. This also helped extract secure and stable session keys with high entropy.

In a series of publications, Maes et al. (2012, 2009) and Maes (2013) discussed seven silicon-based as well as non-silicon-based PUFs. The authors also discussed the secure sketch techniques used to generate session keys that were proposed by Dodis et al. (2004, 2008).

Reviewing the published literature, we can make several conclusions about the current state of the art in using PUFs for IoT authentication and secure exchange:

1) The literature discusses, sometimes implicitly, one algorithm for issuing the CRP pairs: a single challenge is issued and the device response is observed. This is a simple algorithm that does not utilize the IoT device statistical characteristics to its advantage. Perhaps the only advantage of this algorithm is that it maximizes the number of CRP pairs, which is critical, especially for weak PUFs.

2) The parameters that define the response of the IoT PUF device are not identified in most published works. General statements are typically stated such as: “a large number of response bits are needed to differentiate valid from counterfeit devices”. At best a sketch is provided about the desired Hamming distance (HD) separation between valid devices and counterfeit devices.

3) Values of the PUF circuit parameters, the statistical parameters and the choice of overall system parameters are not studied to see how the response of the PUF can be controlled and optimized. A lack of accurate logical PUF models explains why this is the accepted view of using PUFs.

3.3. System design considerations

Telehealth Network Model: Figure 3.3 shows the architecture of the telehealth system. The main agents in the system include:

– **Network Server (S)**: The network server is usually located in a hospital. We can consider the server to be a root-of-trust (RoT) since it contains tamper-resistant hardware like a trusted platform module (TPM).

– **Mobile User (M)**: This can be thought of as the smart devices or telephones used by the healthcare professionals such as doctors and nurses.

– **IoT Edge Device (D)**: The IoT edge devices include Internet-enabled sensors/actuators that could be located in a remote health care unit or could be located in a body area network (BAN) attached to a stay-at-home patient.
3.4. Silicon physically unclonable functions (PUF)

Silicon static random access memory (SRAM) used to construct a PUF is a practical technique to give a unique “fingerprint” or identity to a silicon device and the ability to generate a secret key without the need to store it in NVRAM. The main advantages of silicon SRAM PUF are several:

1) Silicon SRAM based on CMOS technology does not require any extra processing steps which makes them practical to implement at no additional costs or delays (Holcomb et al. 2009).

2) The area cost is less than that required by an identity stored in NVRAM since circuits often require extra hardware such as charge pump to program the NVRAM.

3) The identity cannot be cloned or reverse-engineered without destroying the fingerprint itself and removing the possibility of any device recycling.

4) The number of CRP goes beyond the number of words of the memory. In fact, the number of challenge-response pairs (CRP) is given by equation [3.6] or equation [3.7] later in this chapter.

In addition, a PUF provides tamper resistance since any changes to the device physical parameters will lead to a corrupted identity (Maes et al. 2009). The concept of silicon PUF was first proposed by Gassend et al. (2002). Silicon PUF operation relies on the inevitable random variations that are introduced during the fabrication of semiconductor devices. This gives the means to uniquely identify the individual.
devices. Furthermore, such a PUF can not be replicated through reverse engineering even by the device manufacturer. SRAM cells provide an compact way to create a silicon PUF through the unique startup values of the individual words in the memory (Guajardo et al. 2007; Boehm and Hofer 2009; Schrijen 2020). The SRAM content each time the SRAM PUF starts up is slightly different due to the inevitable dynamic noise (Su et al. 2008; Yu et al. 2011). Dodis was the first to propose using forward error correcting codes (FEC) to overcome the noisy inconsistent SRAM PUF output (Dodis et al. 2004, 2008). This was later improved upon by other authors (Boyen 2004; Bösch et al. 2008; Maes et al. 2009, 2012; van der Leest et al. 2012; Maes 2013; Delvaux et al. 2014; Hiller 2016; Delvaux 2017a, 2017b; Gao et al. 2018, 2019; Schrijen 2020).

3.4.1. Mutual authentication and key exchange using PUF

Figure 3.2 shows the basic structure of the secure sketch at the server and client. The server selects a challenge $c$ and uses the database supplied by the manufacturer to extract the expected response $r$. The server also performs forward error correction coding (FEC) on the response to produce helper data $w$. The secure sketch also produces a hashed value $h$ for the response. This value will serve to establish mutual authentication between the server (gateway provided by Internet service provider, in our case) and the client (IoT edge device, in our case).

![Figure 3.2. Basic structure of the secure sketch at the server (on the left) and client (on the right). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip](www.iste.co.uk/khatoun/cybersecurity.zip)

The client receives the challenge $c$ and helper data $w$ and in response produces the actual noisy response $r'$, and with the help of $w$ it decodes $r'$ to produce the error-free response $r$. The client then hashes this value and sends $h^*$ to the server to be authenticated.
3.4.2. Fuzzy extractor

At the server side, the fuzzy extractor uses the expected response \( w \) to generate the secret key \( K \) and helper data \( w \) as shown on the left in Figure 3.3. The helper data \( r \) can be made public without divulging the secret key. On the right side of Figure 3.3, the IoT device with the PUF is the client which, upon receiving the challenge \( c \) and helper data \( P \), generates the noisy response \( r' \). As long as the Hamming distance between \( r \) and \( r' \) is less than a certain threshold, the fuzzy extractor uses the corrected response \( r \) and helper data \( w \) to generate the secret key \( K \).

![Figure 3.3. Basic structure of the fuzzy extractor at the server and client. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip](image)

It should be noted that the secret key changes each time a new challenge \( c \) is issued. In this chapter we will use this feature to generate a nonce which could be \( K \) or a hashed value of \( K \) to increase its entropy. This will serve to construct a secret key shared among the entities of our system: mobile device (\( M \)), server (\( S \)), and IoT edge device (\( D \)).

The key regeneration using the fuzzy extractor process can be expressed by the equation

\[
(K_d, N_d) = \text{key\_regen}(c, w)
\]  

[3.1]

where \( K_d \) is the secret key and \( N_d \) is the secret random number.

Some implementations were done in FPGA platforms (Herrewege et al. 2012; Maes et al. 2012) and some were implemented on microcontrollers (Aysu et al. 2015). Gao et al. (2019) proposed an SRAM-based PUF key generator on a microcontroller using RF energy harvesting.

3.5. Convolutional encoding and Viterbi decoding the SRAM words

As explained in section 3.4.1, the PUF response is inherently noisy due to the CMOS dynamic noise. Means have to be provided for removing this noise from the response. This is the job of the secure sketch which is derived from forward error correcting coding (FEC) theory. The helper data \( w \) in Figure 3.2 is used to remove the
dynamic noise. However, the system designer must ensure that $w$ does not reveal any information about the device response since $w$ will be sent across unsecured channels. Furthermore, the error correcting capability of the secure sketch must be limited to a certain number of bit errors. If it exceeds that limit, there is a danger of inadvertently converting the response from a counterfeit device to that of a valid device.

Convolutional codes are a powerful FEC technique that is the only FEC that can handle both random errors and burst errors. The error correcting capability can be increased or decreased by increasing or decreasing the code rate, respectively.

A rate $k/n$ convolutional encoder accepts $k$ message bits and adds redundant bits to produce $n$ output bits for each message with $n > k$. A convolutional encoder is specified by the three-tuple $(n, k, m)$ where:

1) $n$: number of bits of the message after encoding;
2) $k$: number of information bits of the message before encoding;
3) $m$: order of the code or number of storage registers.

The code rate is defined by the first two parameters $k/n$. We can write the convolutional encoder as

$$y_i[n] = \sum_{j=0}^{k-1} h_i[j] x[n-j]$$

where $x[n]$ is the $k$-bit input symbol, $y_i[n]$ is the $i$th $k$-bit output symbol with $0 \leq i < n$, and $h_i[j]$ is the $k$-bit generator polynomial weight with $0 \leq j < m$ the number of delay elements. $x[n]$ represents the input symbol stream and $y[n]$ represents the output symbol stream.

Figure 3.4 shows a 1/2 convolutional encoder. The disadvantage of the structure in Figure 3.4 is the delay incurred to add $m$ inputs using XOR gates. The constraint length for the $k/n$ encoder with the structure in Figure 3.4 is $km$, which indicates the number of delay elements needed to generate the outputs from $k$ inputs.

Figure 3.5 shows an alternative form for a 1/2 convolutional encoder. This form has the advantage of pipelining the partial output results at the expense of doubling the number of delay elements. The constraint length for the $k/n$ encoder with the structure in Figure 3.5 is $knm$, which indicates the number of delay elements needed to generate the outputs from $k$ inputs.

The generator polynomial for the structure in Figure 3.4 or 3.5 is defined as

$$p_1(x) = h_1[0]x^3 + h_1[1]x^2 + h_1[2]x + h_1[3]$$

$$p_2(x) = h_2[0]x^3 + h_2[1]x^2 + h_2[2]x + h_2[3]$$
Typically the generator polynomial is represented in matrix form as an $n \times (m+1)$ matrix $\mathbf{G}$. For the $(2, 1, 3)$ encoder we can write

$$
\mathbf{G} = \begin{bmatrix}
\end{bmatrix}
$$

[3.5]

For a degree 3 polynomial in GF(2) we can use several primitive polynomials such as

$$p(x) = \begin{cases} 
x^3 + x^2 + x + 1 \\
x^3 + x + 1
\end{cases}$$
The golden SRAM words are defined according to the rules in [3.18]. These criteria will determine which bits of a given SRAM word are to be used for encoding/decoding and which ones will be overpassed according to the algorithms discussed in section 3.7.

The manufacturer applies convolutional coding to the golden SRAM data word before transmission. The ICs in the field use Viterbi decoding on the actual PUF output to generate the corrected SRAM word. The decoder uses hard decision algorithm, where each bit is interpreted as either ‘0’ or ‘1’.

3.6. CMOS SRAM PUF construction

The basic structure of an SRAM CMOS cell is shown in Figure 3.6, which is basically two cross-coupled CMOS inverters. An excellent discussion of the operation of the CMOS SRAM memory is found in (Prince 1991, section 5.5). As we shall see in section 3.6.2, part of the requirement for an SRAM PUF is to perform repeated resetting of the SRAM. An SRAM PUF might have to be reset over 1,000 or more times to obtain a dependable response free of dynamic noise. This is a basic feature of the proposed algorithm discussed in sections 3.7.2 and 3.7.3 later in this chapter. The basic SRAM can be reset in one of two ways

1) Disconnect then reconnect the power supply $V_{DD}$. This will force the initial state of the two outputs of the cell to be 0 simultaneously. However, this is a slow process since the power supply rails usually have very large parasitic capacitances.

2) Ground the bit lines $B = \overline{B} = 0$ and set the word line $W = 1$. This will ensure the initial state of the two outputs of the cell to be 0 simultaneously. This option requires modifying the word lines $W$ and $\overline{W}$ for the entire SRAM module. This approach is not feasible if the SRAM block is used to store data in addition to the PUF function.

A third alternative is to modify the cell structure so that resetting the cell can be done at a speed matching the write speed of an SRAM. Figure 3.7 shows the basic cell
structure of a NOR gate-based SRAM PUF and Figure 3.8 shows the details of the cell structure. In Figures 3.7 and 3.8, the contents of the cell are obtained through the bit lines $B$ and $\overline{B}$ for the bit value and its complement, respectively. Signal $W$ is usually referred to as the word line and, when asserted, connects the outputs of the cell to the bit lines. Finally, signal $R$ is the reset signal and when it is asserted to ‘1’, both NOR gates’ outputs will be 0. As soon as $R = 0$, the storage cell stores ‘1’ or ‘0’ depending on several factors such as:

1) Threshold voltage values for the n-MOS and p-MOS transistors of the NOR gates and the pass-gate controlled by signal $W$.

2) Delay between the signal $R$ and the lower inputs to the NOR gates.

3) Parasitic capacitances seen by the outputs of the two NOR gates.

![Figure 3.7. Basic cell structure for NOR gate-based SRAM PUF](image1)

![Figure 3.8. Detail of the basic cell structure for NOR gate-based SRAM PUF](image2)

The cell structure in Figure 3.8 was first simulated using the analog device simulator QUCS (Jahn and Borrás 2007). The simulator confirmed the basic operation of the SRAM cell under normal operation when $R = 0$ and $W = 1$. When the cell was reset ($R = 1$), both outputs $B$ and $\overline{B}$ both reached the same reset value due to symmetry conditions. When the reset was not asserted $R = 0$, the SRAM cell assumed a random value 1 or 0.
It should be mentioned that the cell design can use two NOR gates or two inverters. The inverter-based design, also known as the 6-transistor design, must add enough pass-gates to allow for breaking up the feedback path and setting the inputs of the two inverters to equal values, whether 0 or 1. There is therefore no saving in terms of the MOS transistor count to using the 6-transistor cell design.

Assuming the number of words in the SRAM PUF to be $N$ and that a challenge selects addresses of $k$ words, the number of challenge-response pairs (CRP) is given by the permutation

$$\text{CRP} = N^k \gg N$$

when repetitions are allowed. Alternatively we have

$$\text{CRP} = \frac{N!}{(N-k)!} \gg N$$

when repetitions are not allowed. Adopting this strategy, one can construct strong PUF out of NOR-based SRAM PUF, especially if the order of the response bits is pre-arranged and can be securely varied at the start of each session.

### 3.6.1. SRAM PUF statistical model

The operation of SRAM PUF relies to two random physical phenomena: random processing variations and dynamic noise, which are analog processes. Both these phenomena control the digital binary value of the stored bits after SRAM initialization. Random process variation is static for a given device and facilitates creation of the device “biometric” or unique fingerprint. Random dynamic noise, on the other hand, is dynamic and introduces noise to the device identity (ID).

One way to analyze an SRAM-based PUF is to accurately model the devices and wire delays of the basic cell. However, this will not account for all the factors, such as doping variations, oxide thickness variations, random parasitic loading capacitances, etc. Instead we resort here to developing a logical model that encompasses all these physical phenomena. This approach is akin to the logical modeling of faults instead of modeling all possible physical faults in an integrated circuit.

The random variable we choose to model should be amenable to measurements under mass production settings by the device manufacturer. In the context of using an SRAM PUF, an appropriate random variable is the content of the SRAM memory cells. This is a binary random variable that is characterized by the two probabilities $a$ and $b$ denoting the probability that the SRAM cell is ‘1’ or ‘0’, respectively. Ideally random process variations and CMOS noise are absent and the structure of each
SRAM Physically Unclonable Functions for Smart Home IoT Telehealth Environments

SRAM cell is completely symmetric making the ideal probabilities \( a_i \) and \( b_i \) satisfy the equality

\[
a_i = b_i = 0.5
\]

Due to the central limit theorem, the random process variation (RPV) effect on the pair \((a_i, b_i)\) follows the biased Gaussian distribution whose pdf is given by

\[
f_{A_p} = \frac{1}{\sigma_p \sqrt{2\pi}} e^{-\frac{(a_p-a_i)^2}{2\sigma_p^2}} \tag{3.8}
\]

where \( a_p \) is the adjusted value of \( a_i \) due to RPV and \( \sigma_p^2 \) is the variance of the RPV process. We should note that \( a_i \) and \( \sigma_p \) are identical for all SRAM bits within a device or among different devices.

The value of \( a_p \) is given by

\[
a_p = G(a_i, \sigma_p) \tag{3.9}
\]

where \( G(a_i, \sigma_p) \) is a Gaussian random process with mean \( a_i \) and variance \( \sigma_p^2 \).

Figure 3.9 shows the different types of distributions due to the random processes involved in determining the bit value probabilities.

Figure 3.9(a) shows the pdf of the random variable \( a_p \) due to RPV which is a biased Gaussian process with mean \( a_i \) and variance \( \sigma_p^2 \).

There are several sources of dynamic or short-term noise in CMOS devices including:

1) Thermal noise as additive white Gaussian noise (AWGN), showing flat spectral distribution.

2) Shot noise due to charge carrier flow across \( p-n \) junctions.

3) Flicker noise due to charge trapping in the device, showing \( 1/f \) spectral distribution.

These noise sources introduce variations in the value of transition probability \( a \) each time the CMOS inverter undergoes a transition.

Figure 3.9(b) shows the pdf of the random dynamic noise \( n \) which is given by

\[
f_{A_n}(a_n) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{a_n^2}{2\sigma_n^2}} \tag{3.10}
\]

where \( \sigma_n^2 \) is the variance of the dynamic noise process. On the other hand, the pdf for the additive white Gaussian noise (AWGN) is common to all bits within a device and also for all devices.
Figure 3.9. The pdf distributions of transition probability $a$ due to the different physical phenomena. (a) pdf of $a_p$ due to random process variations (RPV). (b) pdf of $n$ due to random dynamic noise. (c) pdf of $a$ due to the combined effects of RPV and random dynamic noise when $a_p > a_i$. (d) pdf of $a$ due to the combined effects of RPV and random dynamic noise when $a_p < a_i$.

The combined effects of RPV and dynamic CMOS noise generate a pdf given by

$$f_A(a) = \frac{1}{\sigma_n \sqrt{2\pi}} e^{-\frac{(a-a_p)^2}{2\sigma^2_n}}$$

[3.11]

where $a_p$ is the contribution of RPV and $\sigma_n$ is the contribution of random dynamic noise.

Figure 3.9(c) shows the pdf of the transition probability $a$ when both RPF and dynamic noise are present and the mean value $\mu_p > \mu_i$. Figure 3.9(d) shows the pdf
of the transition probability \( a \) when both RPF and dynamic noise are present and the mean value \( \mu_p < \mu_i \). For either case, the probability \( a \) is given by:

\[
a = G(a_p, \sigma_n)
\]  

3.6.2. Extracting the SRAM cell statistical parameters

The value of the a bit at location \( b \) in word \( w \) is denoted by \( v(w, b) \) with \( w \) denoting the SRAM word and \( b \) denoting the location \( b \) in the word. The range of the indices \( w \) and \( b \) is given by

\[
0 \leq w < W \quad \text{and} \quad 0 \leq b < B
\]

where \( W \) is the total number of words in the SRAM and \( B \) is the word size.

The values of \( W \) and \( B \) are set during the fabrication phase of the device. The values of the probability \( a_p \) and variance \( \sigma_p^2 \) can be extracted by the manufacturer during the pre-deployment phase by following these steps:

1) The manufacturer performs \( N \) initializations and observes the stored values of \( v_k(w, b) \) in the tagged bit at each step \( k \).

2) The probability \( a_p \) is obtained as

\[
a_p(w, b) = \frac{1}{N} \sum_{k=0}^{N-1} v_k(w, b)
\]  

3) The variance \( \sigma_n^2 \) due to dynamic noise is obtained as

\[
\sigma_n^2(w, b) = \frac{1}{N} \sum_{k=0}^{N-1} [v_k(w, b) - a_p(w, b)]^2
\]  

Alternatively, the overall \( \sigma_n^2 \) can be estimated as

\[
\sigma_n^2 = \frac{1}{WB} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} \sum_{k=0}^{N-1} [v_k(w, b) - a_p(w, b)]^2
\]  

In order to measure the RPV parameters \( a_i \) and \( \sigma_i \) the manufacturer now studies the contents of all the bits in the SRAM memory.

1) The values \( a_p(w, b) \) for all bits in the SRAM memory are obtained previously.

2) The value \( a_i \) is obtained as:

\[
a_i = \frac{1}{WB} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} a_p(w, b)
\]
3) The value $\sigma_i$ is obtained as:

$$\sigma_i^2 = \frac{1}{WB} \sum_{w=0}^{W-1} \sum_{b=0}^{B-1} (a_p(w, b) - a_i)^2$$  \[3.17\]

### 3.6.3. Obtaining the golden SRAM PUF memory content

The manufacturer of the SRAM PUF can run $N$ initialization steps on each device to obtain the values of $a_i$, $\sigma_i$, $a_p$, and $\sigma_n$, as explained in section 3.6.2. The digitization, or analog-to-digital conversion, step gives the golden or reference stored value $v(w, b)$ of each bit in the SRAM PUF where $0 \leq w < W$ is the SRAM row index or word address and $0 \leq b < B$ is the bit index within a word. The assignment of golden or reference memory content is given by the rules:

$$v(w, b) = \begin{cases} 
0, & 0 \leq a_p(w, b) \leq a_i \\
1, & a_i < a_p(w, b) \leq 1 
\end{cases} \quad \[3.18\]

The conditions in equation [3.18] indicate the cell is skewed toward 0 or 1, respectively, and the effect of dynamic noise is negligible. Such cells provide the desired randomness that make the PUF hard to clone or reverse engineer (Holcomb et al. 2009).

The manufacturer now prepares a dataset for each device’s SRAM PUF. The dataset contains the following information:

1) $W$ the number of words in the SRAM PUF.
2) $B$ the number of bits in each word.
3) Golden value $v(w, b)$ associated with each bit in the SRAM PUF based on criteria in equation [3.18].

The user now has the ability to choose the challenge/response pairs to use.

### 3.6.4. Bit error rate (BER)

The bit error rate of an SRAM cell is due to two mutually exclusive events: the bit is measured as ‘0’ when it should be ‘1’ or it is measured as ‘1’ when it should be ‘0’. We can write the BER as

$$p_e = A + B$$  \[3.19\]

where $A$ is the probability that the SRAM bit is measured ‘0’ when it should be ‘1’ because $a_p > a_i$ and $B$ is the probability that the SRAM bit is measured ‘1’ when it should be ‘0’ because $a_p < a_i$. The two probabilities are

$$A = \int_{a=0}^{a_i} \frac{1}{\sigma_n \sqrt{2\pi}} e^{-(a-a_p)^2/2\sigma_n^2} da$$  \[3.20\]
for the case when \( a_p > a_i \), and

\[
B = 1 - \int_{a=0}^{a_i} \frac{1}{\sigma_n \sqrt{2\pi}} e^{-(a-a_p)^2/2\sigma_n^2} \, da
\]

[3.21]

for the case when \( a_p < a_i \).

### 3.6.5. Signal-to-noise ratio (SNR) for SRAM PUF

The term “signal” in the context of this work refers to the probability \( a_p \). More specifically, we take the absolute difference \( |a_p - a_i| \) as the definition of our signal for the following reasons:

1) When \( a_p = a_i \) the SRAM cell value has equal probability of being 1 or 0 and this value totally depends on the effects of dynamic noise.

2) When \( a_i < a_p \leq 1 \) the SRAM cell value is biased to be 1 with little effects from dynamic noise especially when \( a_p \to 1 \).

3) When \( 0 \leq \mu_p < a_i \) the SRAM cell value is biased to be 0 with little effects from dynamic noise especially when \( a_p \to 0 \).

We can now define the system-level signal-to-noise ratio (SNR) of a tagged SRAM cell as the ratio of the energy due random process variations relative to dynamic noise energy:

\[
SNR = 10 \log \left( \frac{(a_p - a_i)^2 + \sigma_p^2}{\sigma_n^2} \right)
\]

[3.22]

where the contribution of the random process variations (through \( a_p \) and \( \sigma_p \)) and dynamic noise (through \( \sigma_n \)) are beyond the control of the device manufacturer.

Bits in an SRAM word, and for that matter, all bits in the SRAM, do not have the same SNR. The minimum SNR is when \( \mu_p = a_i \):

\[
SNR_{\text{min}} = 10 \log \left( \frac{\sigma_p^2}{\sigma_n^2} \right)
\]

\[= 20 \log \left( \frac{\sigma_p}{\sigma_n} \right) \]

[3.23]

On the other hand, maximum SNR occurs when either \( a_p = 0 \) or when \( a_p = 1 \). Since \( a_i = 0.5 \), we can write:

\[
SNR_{\text{max}} = 10 \log \left( \frac{a_i^2 + \sigma_p^2}{\sigma_n^2} \right)
\]

[3.24]

When \( SNR \approx SNR_{\text{min}} \), the response to the challenge is noisy. Similarly when \( SNR \approx SNR_{\text{max}} \), the response to the challenge is more stable and less dependent on noise.
3.7. Algorithms for issuing CRP

In this section we propose and analyze several algorithms for issuing the CRP data and their effect on system design.

3.7.1. Algorithm #1: single-challenge

The single-challenge algorithm used to authenticate a device follows the steps depicted in Figure 3.10. Four steps are required for authenticating the device and generating the session key.

# 1: Server selects a single CRP \((c, r)\)

# 2: Server generates \(w, K, h\)

# 3: Client uses \((c, w)\) to generate \(r'_1, K, h^*\)

# 4: Server authenticates device

<table>
<thead>
<tr>
<th>Server</th>
<th>Channel</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td># 1. Select CRP ((c, r))</td>
<td># 2. Generate (w, K, h)</td>
<td># 3. Use ((c, w)) to generate (r'_1, K, h^*)</td>
</tr>
<tr>
<td># 4. Verify (h^* = h)</td>
<td>(\leftarrow h^*)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.10. Algorithm #1 for the authentication of an IoT edge device and secure key exchange

Table 3.1 shows the maximum intra Hamming distance and inter Hamming distance for different word sizes \(B\) for the case when \(W = 1K\) words, \(N = 1024\) initialization operations and \(SNR_{max} = 20\) dB.

We observe from Table 3.1 that the number of errors in the PUF response increases as \(B\) increases, as indicated by the intra Hamming distance. The errors are due to the effects of dynamic noise. We also observe from Table 3.1 that word lengths \(B \geq 256\) are required to ensure clear separation between different device IDs.
Algorithm #1 is vulnerable to effects of dynamic noise which leads to a large intra Hamming distance and a small, or even negative, inter Hamming distance. The former leads to developing error correction codes capable of correcting a large number of bits. The latter might lead to false positive that declares or accepts a device as being authentic while it is, in fact, fake.

To be able to mitigate the above effects, the system designer must be able to ensure that the distribution of the intra Hamming distance is sufficiently separated from the inter Hamming distance. This approach is expensive since it requires:

1) Using large SRAM word size.

2) Being able to correct a large number of error bits through using many redundancy bits.

Table 3.1. The Algorithm #1 maximum intra Hamming distance and inter Hamming distance for the case when \( W = 1\)K words, \( N = 1024 \) initialization operations and \( SNR_{max} = 20 \) dB

<table>
<thead>
<tr>
<th>( B ) (bits)</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Intra Hamming Distance (bits)</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>92</td>
<td>171</td>
</tr>
<tr>
<td>Inter-Intra Hamming Distance Separation (bits)</td>
<td>-12</td>
<td>-6</td>
<td>-3</td>
<td>10</td>
<td>49</td>
</tr>
</tbody>
</table>

Figure 3.11 shows the histograms for intra and inter Hamming distance distributions for the case when \( W = 1\)K words, \( B = 128 \) bits, \( N = 1024 \) initialization operations and \( SNR_{max} = 20 \) dB. We notice that when \( B = 128 \) bits the inter and intra Hamming distance histograms are touching. It would be hard to distinguish between a valid device and a fake one.

Figure 3.12 shows the histograms for intra and inter Hamming distance distributions for the case when \( W = 1\)K words, \( B = 512 \) bits, \( N = 1024 \) initialization operations and \( SNR_{max} = 20 \) dB. When \( B = 256 \) the the inter and intra Hamming distance histograms are well separated. It would be easy to distinguish between a valid device and a fake one.

Figure 3.13 shows the histograms for intra and inter Hamming distance distributions for the case when \( W = 1\)K words and \( B = 512 \) bits. When \( B = 512 \) the separation between inter and intra Hamming distances is increased compared to the case when \( B = 256 \).

It might prove expensive to implement an SRAM PUF with a word size of 512 bits. This problem can be simply solved by changing the challenge \( c \) to use multiple words that need not be consecutive. For example, if the SRAM PUF is a memory with word size \( B = 64 \), then generating a 512-bit response is feasible by simply having the challenge \( c \) correspond to addressing 8 words. This actually allows us to enrich the
space of possible challenges by being able to generate all possible permutations so that we have $8! = 40,032$ possible challenges that use the same 8 words of the SRAM.

**Figure 3.11.** The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when $W = 1K$ words, $B = 128$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB

**Figure 3.12.** The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when $W = 1K$ words, $B = 256$ bits, $N = 1024$ initialization operations and $SNR_{max} = 20$ dB
Figure 3.13. The Algorithm #1 histogram on the left shows the intra Hamming distance distribution. The histogram on the right shows the inter Hamming distance distribution. The case when \( W = 1K \) words, \( B = 0.5K \) bits, \( N = 1024 \) initialization operations and \( \text{SNR}_{\text{max}} = 20 \) dB

3.7.2. Algorithm #2: repeated challenge

The basic idea behind Algorithm #2 is to eliminate dynamic noise by repeating the steps used by the manufacturer to obtain the golden reference SRAM as discussed in section 3.6.2.

Algorithm #2 performs \( N \) initializations of the SRAM PUF and prepares an \( N \times B \) response matrix \( R' \) whose rows are the individual responses \( r'[n] \) for the same challenge \( c \). A row vector \( x \) is obtained as the sum of columns of \( R' \):

\[
x = \frac{1}{N} \times \text{SumColumns}(R')
\]  

[3.25]

where \( \text{SumColumns}(R') \) sums the individual \( B \) columns of matrix \( R' \) to produce a row \( B \)-vector \( x \). The sum operation effectively cancels out the random dynamic noise which effectively performs repetition coding or majority voting.

The response of the device being authenticated is estimated in bitwise fashion. The bit at location \( b \) of \( w'_2 \) is obtained as:

\[
u'_2[b] = \begin{cases} 0 & \text{when } 0 \leq x[b] < a_i \\ 1 & \text{when } a_i \leq x[b] < 1 \end{cases}
\]  

[3.26]

Using the helper data \( w \), the error-corrected response \( r_2 \) is obtained. The steps used by Algorithm #2 are shown in Figure 3.14. Four steps are required for authenticating the device and generating the session key.
# 1: Server selects a CRP \((c, r_2, N)\)

# 2: Server generates \(w, K\) and \(h\)

# 3: Client uses \((c, w, N)\) to generate \(r_2', K\), and \(h^*\)

# 4: Server authenticates device

<table>
<thead>
<tr>
<th>Server</th>
<th>Channel</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td># 1. Select CRP ((c, r_2, N))</td>
<td># 2. Generate (w, K, h)</td>
<td># 3. Use (c, w) to generate (R', r_2', K), and (h^*)</td>
</tr>
<tr>
<td># 4. Verify (h^* = h)</td>
<td>(\leftarrow h^*)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.14. Algorithm #2 for the authentication of an IoT edge device and secure key exchange.

Table 3.2 shows the maximum intra Hamming distance and inter Hamming distance for different word sizes \(B\) for the case when \(W = 1K\) words, \(N = 1024\) initialization operations and \(SNR_{max} = 20\) dB.

<table>
<thead>
<tr>
<th>(B) (bits)</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Intra Hamming Distance (bits)</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>Inter-Intra Hamming Distance Separation (bits)</td>
<td>7</td>
<td>21</td>
<td>42</td>
<td>211</td>
<td>207</td>
</tr>
</tbody>
</table>

Table 3.2. The Algorithm #2 maximum intra Hamming distance and inter Hamming distance for the case when \(W = 1K\) words, \(N = 1024\) initialization operations and \(SNR_{max} = 20\) dB.

We see from the table that the intra Hamming distance is at least an order of magnitude less than the case for Algorithm #1. The inter Hamming distance, of course, remained the same as in Algorithm #1.

From Table 3.2 we make a very interesting discovery which is the ability to reduce the word size \(B\) and yet be able to authenticate devices. Table 3.2 shows that we are able to authenticate IoT devices even when \(B \approx 32\) bits. This would not be possible with Algorithm #1.

3.7.3. Algorithm #3: repeated challenge with bit selection

Algorithm #3 is derived from Algorithm #2. The main idea of this algorithm is to consider or select the response bits that have high SNR in a further attempt to reduce
the effects of dynamic noise. The criterion to select a response bit to be part of the filtered response is given by

\[
w'_3[b] = w'_2[b] \quad \text{when} \quad \begin{cases} 
0 \leq x[b] < a_i - \Delta \\
\text{or} \\
a_i + \Delta < x[b] \leq 1
\end{cases}
\]

The steps used by Algorithm #3 are shown in Figure 3.15 where \( A \) is the vector of bit addresses selected according to equation [3.27]. Four steps are required for authenticating the device and generating the session key.

# 1: Server selects a CRP \( (c, r_3, N, A, \Delta) \)

# 2: Server generates \( w, K \) and \( h \)

# 3: Client uses \( (c, w, N, A, \Delta) \) to generate \( r'_3, K, \) and \( h^* \)

# 4: Server authenticates device

![Figure 3.15. Algorithm #3 for the authentication of an IoT edge device and secure key exchange.](image)

Table 3.3 shows that we are able to authenticate IoT devices even when \( B \approx 32 \) bits. This would not be possible with Algorithm #1.

<table>
<thead>
<tr>
<th>( B ) (bits)</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum Intra Hamming Distance (bits)</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Inter-Intra Hamming Distance Separation (bits)</td>
<td>7</td>
<td>21</td>
<td>40</td>
<td>95</td>
<td>213</td>
</tr>
</tbody>
</table>

**Table 3.3.** The Algorithm #3 maximum intra Hamming distance and inter Hamming distance for different word sizes \( B \) for the case when \( W = 1K \) words, \( N = 1024 \) initialization operations, \( SNR_{max} = 20 \) dB and \( \Delta = 0.3 \).
3.8. Security of PUF-based IoT devices

The smart home is the target of several attacks such as (Fakroon et al. 2020):
1) replay;
2) eavesdropping;
3) device loss;
4) impersonation;
5) man-in-the-middle;
6) forward-backward secrecy;
7) user credentials;
8) session key guessing;
9) user identification and tracking;
10) side-channel;
11) over-production and counterfeiting;
12) deep learning and machine learning;
13) reverse engineering;
14) nonvolatile memory attacks.

We should note several general principles to ensure the security of the telehealth system, which includes smart home and IoT devices.

– The attacks mentioned above depend on getting the secret key associated with each IoT device through targeting the NVRAM content. Here this is prevented through storing the secret keys within the circuit structure of the PUF.

– Studying the CRP responses is thwarted by hiding the IoT device response \( r \) and never sending it between the communicating entities. This provides a level of protection against using deep learning to mimic the PUF function.

– Secret session keys \( K \) and hash values \( h \) are based on chaining and context such that a previous hash value or current device environment are used to generate a session \( K \) and \( h \) in addition to the response \( r \) (Fakroon et al. 2021).

– The use of PUFs in IoT devices constitutes an inexpensive means of providing tamper-proofing to a certain degree. It is not expected that each IoT device would be a root of trust (RoT) but at least it provides immunity to reverse engineering and tampering.

– Security measures must be layered starting from the physical layer (PUFs), then the communication layer and ending with the application layer. Multifactor authentication is also feasible here since each PUF can provide some of these factors.
3.9. Conclusions

This chapter developed novel statistical models for SRAM PUF performance. The main parameters affecting the SRAM PUF performance were identified and techniques to measure them were proposed. These parameters can be estimated by the manufacturer at the pre-deployment phase and can also be measured in the field. This chapter also proposed three algorithms for generating CRP and establishing device authentication and secure key exchange. Algorithm #1 is based on a single challenge. Algorithm #2 is based on repeated challenge. Algorithm #3 is based on repeated challenge with bit selection. We noted that Algorithm #1 can be used when the SRAM word size is $B > 256$ bits. Further, Algorithm #1 introduces a rather large number of bits in error in the response. Two new algorithms are proposed in this chapter: Algorithm #2 and Algorithm #3. These two algorithms solved the two main problems associated with noisy PUF responses: the need to use a large number of bits $B$ and the large number of errors in the response.

3.10. Acknowledgements

This research was supported by a grant from the National Research Council of Canada (NRC) through the Collaborative R&D Initiative.

3.11. References


complete control of security. Thus the growth of secure IoT systems with lightweight security apps will resolve the most important requirements in the future.

Technology has changed the community conversation on buying and selling products, education, medication, farming, as well as the security. Consequently, IoT based smart home systems contain inbuilt security to combat the threats. Hence the numerous wireless network solutions, such as wireless Ethernet, ultra wide band (UWB), Bluetooth and others related to the field of home networking come into play (Yashwant et al. 2020). In view of the fact that Bluetooth is now widespread in mobile devices, it provides a cost-effective and secure solution to support the wireless network for IoT-based home systems. With increased growth in smart homes and improvements to a considerable number of manufactured smart home goods on the market, home security has become extremely fashionable (Manyika et al. 2013). Additionally, the prototype implementation system would be an excellent solution to support access monitoring and the control system for smart home security in IoT. A number of scenarios are involved in the development of communications procedures for the purpose of smart home security, leveraging standard procedures and encoding the interfaces, along with providing automation platforms that reduce the number of threats within dissimilar computing platforms and appliances (Drahansky et al. 2016). Consequently, smart homes contain appliances such as refrigerators, thermostats, home security, self-operating vacuum cleaners, clean-up and safeguarding devices, as well cameras, light sensors and motion sensors, which also collect data and information (Drahansky et al. 2016). A large amount of confidential information and susceptible data exists, such as: addresses, locations, representations and network related information. The information can be available to the device manufacturer, mobile application holder, and intermediary dealers or the community, depending on their rights. There are small numbers of offerings that deal with confidentiality in the environment of smart home security (Kraemer et al. 2018).

However, numerous modification techniques are in place to analyze the confidentiality and security associated with IoT-related information processing and threats (Almusaylim and Zaman 2019). Considering the publicly authorized aspects of security in IoT related areas, government associations are becoming heavily involved in IoT based security and interoperability from a standards perspective. When customers obtain IoT-based smart home devices openly from service providers, they may possibly work with the extremely insignificant knowledge while approving the terms and conditions (Ts and Cs), and privacy policy (PP). A significant number of standards have been specified by the Open Connectivity Foundation (OCF). For most of the standards, interoperability is key to increasing security in smart homes and allowing customers and commercial organizations to
communicate (Philippe et al. 2011). With the growing needs, the IoT marketplace is witnessing an increase in the technology and tools that would support the reliable security approach (ENISA 2017). The most important technological concerns include:

– **Regulations:** Government directives protect the computer systems, as well as other information technology components, with the intention of improving security. A lot of associations and companies pitch for regulations to secure the information systems from cyber-attacks and governments issue regulations. For example, NERC-CIP for power efficiency in North America, and also instruction on security of Network Information Systems (NIS Directive) in Europe. Associations need to follow the regulations to plan the new devices and processes.

– **Standards:** Facts and reports on how certain approaches should be relayed in a reliable way are very important. Methods are usually standardized in a published resource, in an attempt to improve the cyber security and cyber surroundings of a consumer or association. The main concern is reducing the risk, together with preventing and mitigating the attacks. Conforming to standards is regularly required to make sure that a satisfactory level of excellence is achieved. For instance, the IEC-62443 standard is well regarded in the control of industrial and smart home systems.

– **Guidelines:** Detailed instructions on the usage and implementation of techniques are required to protect smart home systems and their surroundings. Guidelines are related to standards and need measured references and approval from the stakeholders. For instance, the NISTIR 7628 guidelines are in place to support smart grid-based home security (Pillitteri 2014).

– **Policies:** Guiding principles and procedures meet the security requirements of smart homes, organizations, as well as the society at large. Compliance with the policies has become compulsory. Policies offer top-down requests in support of smart home, business and commerce, in order to secure the information and assets. They work with the set of laws, regulation and authorized requests.

– **Procedures:** In order to apply standards and policies, detailed procedures need to be defined. Procedures contain sequences of detailed steps and instructions that should be followed in order to achieve a target. These become mandatory to illustrate and guide the end users. For instance, the maximum amount of time for a user account password to become obsolete and indicating the new requests to be produced.

It is important to note that standards are not yet mandatory at the time of implementing and designing the systems and platforms of IoT based smart home
security. However, the general procedure should become easier and foster the development of a capacity to establish innovative tools and techniques (Brass et al. 2018). Similar application-related techniques on security may include facial recognition, fingerprint scanning and password protection. These applications observe the behavior of customers and consequently improve security. The amalgamation of technologies such as machine learning, omnipresent wireless communications, embedded systems, and real-time analytics has led to the development of new applications with IoT potential to support a large number of domains (Hoque et al. 2019).

Latent IoT applications include the remote based monitoring and control of smart home devices, energy consumption and managing smart light and smart lock type devices (Ahmed et al. 2016; Davidson et al. 2018; Hoque et al. 2019). For example, the customer details needed to access the smart lock must be mentioned and stored within the server, together with a time and date schedule. This schedule is able to further utilize to foresee the date and time when the customer will come into the house and handle the security. When the door is locked, the lights should turn off automatically, and when the door is unlocked, the lights should turn on automatically. Anyone can set holidays or vacations on the devices and the system will be on the highest security until the customer returns. Users are also able to set provisional keys in support of the household or in favor of visitors (Davidson et al. 2018). A smart home security system design has been presented in Figure 4.2. This kind of smart home security system can be very cost effective (Hoque et al. 2019). The procedures that need to be used to control and monitor each component of a smart home include the following.

- **Control and monitor components:**
  - **Android application:** This application makes the interface between the customer and smart home devices available, such as locks. This application is used to control and monitor each component of a smart home system.
  - **Web server database:** This is used to keep records of customer based activities. In addition, anyone is allowed to access a device and store the approved person or customer’s identification, for example, their login ID or password.
  - **Control unit:** This becomes the central monitoring unit, used to control and communicate with all of the related components in use within the smart home system. To build a low cost smart home security system, inexpensive components can be used, such as the microcontrollers from Elegoo and the Raspberry PI and RF signals as a communication channel between devices.
– **Window or door component:**

  - **Camera:** This is used to capture the image of the person accessing the lock. The camera is activated with the help of sensors, when someone comes close to the windows or doors.

  - **Motor:** This is a device that can control the latch to open and close.

  - **Fingerprint sensor:** This type of sensor is used to validate the person and offer a protected environment. The use of a fingerprint sensor also presents an organized method of unlocking a door.

  - **Motion sensor:** This type of sensor monitors movement near the doors and windows that can offer access into home. If there is movement in front of these spaces, the camera is activated so that the person can be observed.

– **Smart home component:**

  - **Relay:** This module becomes a part of the hardware device used to support the remote device knobbing. Through this module, the user is able to automatically control the devices using remote access. Appliances can be organized as remote power-driven with on or off commands.

  - **Gas or smoke sensor:** This type of sensor is used to activate an alarm to inform the customer if there is a gas leak in the home.

  - **Light sensor:** This sensor can help in the management of the lighting in the house. For examples, the lights are switched on when the user enters the house (if artificial lighting is required) and they are automatically switched off when the user leaves the home.

– **Smart attentive components:**

  - **Alarm:** This is used to alert the nearby environment in case some urgent attention is required for a fire, gas leak or any other kind of unauthorized access.

  - **GSM module:** This module is a kind of device that can send messages to the customer in any emergency situation. It can make the customer attentive and can provide an exclusive Internet link to access the house remotely.

Along with these components, the system software is designed to include an immediate database that can take care of these different devices. An Android operating system also becomes essential for the development of applications involving XML, Python, JAVA, etc.
Network security continues to be a substantial concern within the smart home community. In order to combat the network attacks, a number of security improvement strategies have been created and distributed within the Internet community. Surprisingly, these attacks are frequently either customized to run off detection or completely new ones to forfeit the plans. Although it may not be possible for the average user to understand, the attacker’s attempt to infiltrate Internet related systems presents the most common and recurrent problem. For this reason, different levels of network security have been established in IoT on the basis of network levels. Customers are obtaining IoT devices for smart homes on the assumption that the producer has embedded suitable security in the devices.
Therefore, in a fundamental approach, the common security threats are identified and resolved at the network level (Sivaraman et al. 2015; Ray et al. 2020).

The concerns over network security have become more complicated as the range of devices has increased past computers such as desktops or laptops. These systems use operating systems (MacOS, Linux, and Windows) with small memories and protection mechanisms. Commonly, these devices are able to connect via peer devices or networks, using similar protocols to that of wireless networks, such as BLE, Bluetooth, NFC, ZigBee, Wi-Fi, LoRaWan, Thread etc. (Gomez and Paradells 2010). The Internet Engineering Task Force (IETF) has put in a strong effort during the establishment of required light weight communication protocols in support of controlled environments, in addition to the existing IP network (Lin and Bergmann, 2016). This contains the IPv6 in excess of Low Power Wireless Personal Area Networks (6LoWPAN: RFC 6282) (Hui and Thubert 2011), the IPv6 Routing Protocol for Low power and Lossy Networks (RPL: RFC 6550) (Winter et al. 2012), along with the Constrained Application Protocol (CoAP: RFC 7252) (Castellani et al. 2017).

The modest instance contains the suitable access control procedures that safeguard the particular IoT device. Whereas a typical instance may engage energetic procedures that modify the access control derived from the situation (for example, the members of family that are present or absent from home). Complicated security similar to those that need a mixture of network management and data analytics becomes absent right now, and is able to satisfy network level security. The network level security has been described in Figure 4.3, where the security management provider interconnects with the Internet service provider or the tools of home router from one side, using dynamic application programming interfaces (APIs), and with home customers using a user-friendly graphical interface from the other side. The purpose of the SMP is to implement the control of configuration in excess of the Internet service provider (ISP) network, as well as the home router for the customer (Dahiya 2017; Lin and Bergmann 2016).

**SMP procedures and benefits:** The security management provider offers apps or portals of customization interfaces to customers, translating these into network level functions summoned by application programming interfaces (APIs). This purposely decouples the security management provider from the infrastructure based dealer or operator, so that multiple objectives are able to compete on behalf of the procedure. The vendor of a home router or ISP could potentially develop SMP abilities in the home, increasing maintenance costs and thus income. A content provider (for example, Netflix or Google) or cloud service provider (for example, Apple or Amazon) may also have an interest in the procedure, in order to develop its
individual services. Otherwise, an innovative competitor could adopt a similar procedure with better analytics and visibility of home network systems. The ease with which SMP procedures can be implemented is a potential asset to business organizations that can help to overcome the existing stagnation in home Internet contributions (Lin and Bergmann 2016).

**Figure 4.3. Network level security**

**Internet service provider and home router vendor procedures and benefits:** Nowadays, home routers (like commercial routers) are included in the network and may have various management interfaces; they may have come from different vendors. These vendors may give up user interface improvement, as well as consent to an outdoor unit of SMP to organize network behavior (the prototype controls open source platforms, for instance OpenWRT). This decreases the growth related problem for vendors, permitting them to concentrate on viable improvement, whereas the model of cloud based control is able to provide the best response to the aspect of device usage. A corresponding argument affects the ISP, which offers Internet connectivity at low-margins. ISPs may monetize the large level of marketplace service customization, and they can also decrease the load of consumer management. The configuration of network APIs can be computerized using software defined network (SDN) technology and consequently, the ISP is able to sustain them at a small price (Lin and Bergmann 2016).

**Embedded devices procedure and benefit:** This system of embedded devices produces and runs a control system to secure the IoT-based smart home. Embedded systems and devices include a wide range of appliances contained by the inhabitant, business, healthcare, automotive, and other types of users. Usually, embedded
systems include firmware or operating systems in support of designed, managed and controlled security related issues in IoT related smart home applications. These devices become smaller in size due to low power consumption, as well as low computing power, such as when monitoring the sensitivity rate. The embedded monitor inside of a timepiece can be attached to an Android device or Smartphone to demonstrate the status of the heart in real time. Automatic Teller Machines (ATM) and Point of Sale (POS) also become models of embedded systems or devices. These numerous features contact the procedures that desire the most excellent control system, together with outlay and complicity of system installation toward the greatest technological outcomes (Gann et al. 1999).

**Virtual or cyber security procedures and benefits:** security is the major concern in positioning IoT-based smart homes. IoT systems make use of wireless communication protocols to provide customers and devices with the ability to drive and take delivery of information involving each other with control and secure IoT-based home appliances. This can build up a complete system that is extremely susceptible toward the attacks of hackers. Consequently, the systems and devices must be installed into the smart home within the security mode, ensuring that these are not observable toward any intruder. A protocol of Near Field Communication (NFC) can also be applied, and this enables the related devices to use peer-to-peer to set-up the communication (Song et al. 2008). Accordingly, they are allowed to transmit data with everyone within extremely close proximity. The mechanism will not disclose the related ID and match up through any Smartphone in anticipation of the customer obtaining the related devices and passing it on the control or knob; then the knob discloses the ID related information and pairs with the customer’s receiver. Though, immediately after the customer has set-up the communication via the system, the customer’s device ID is screened toward the intruders or hackers, consequently increasing security related concerns. This complexity deals with encrypted data on the dispatcher device after decrypting it on the target device and vice versa. The procedure of encryption is utilized based on the communication protocol of machine-to-machine (M2M).

**Customer procedure and benefits:** Customers require the security in support for smart homes to be enhanced further than through the ISP or router vendor only. The main concern of the customer is learned and stored within the cloud, and re-established though the subscriber, which modifies the home router or ISP. The customer should be able to adapt security from the cloud, with the opportunity to modernize the configuration as a tool. ISPs offer management and control of the home gateway, also providing the physical home gateway subscriber or a virtual instance through the cloud (for example, vCPE). This theory becomes important and the resolution is given towards tasks pessimistic of inheritance through the network.
address translator (NAT), allowing smart home gateways (Lin and Bergmann 2016; Dahiya 2017).

**Network controller of the ISP:** The Floodlight (v0.9) OpenFlow controller is used to support the ISP network and improved Java modules are used with RESTful APIs (Lin and Bergmann 2016). The innovative modules connected to FloodLight execute the API to support access control, so that access control procedures (derived from a remote IP) can be pushed via the outdoor SMP unit to support a particular home device.

**SMP Security composer:** A security composer or Ruby-on-Rails is executed, and seizes the position and logic required by the SMP to deal with security in support of the subscriber. The ISP using the aforesaid APIs interrelates with the front-end gateway, as well as customer apps, using RESTful APIs. This is performed on the REST rules or commands from the customer apps or portal. This can recover the suitable state information subsequent toward the command of the subscriber, with the proper series of ISP and APIs toward the attainment of the functionality (Lin and Bergmann 2016).

**Web-based app:** Front-end support is made available to the customers to modify services, as well as execute them in HTML or JavaScript. The customer is able to see and manage it through a web-based app. Customers can see the smart home devices scheduled to work and they can access any support of security locations. The SMP seizes the information depending on the suitable techniques toward securing that exact device, and is able to insert correct access control procedures using the API network, possibly with the perspective of information from the home.

Therefore, the service of enclosing the SMP on condition of the IoT security becomes an added value service with the help of other supporting devices and mechanisms. The Nestsmoke-alarm and Philips Hue lightbulb are good examples of the support devices. The smart light bulb connects to the Internet via Wi-Fi, and iOS or Android apps propel the most wanted commands to regulate the settings of smart bulb. Nowadays, customers would nearly all be aware of this attack, and would know how to block it. The SMP appeals to the API network in the direction of insert suitable access control procedures that enable only identified users to access the smart bulb (Lin et al. 2016; Dahiya 2017). In terms of roaming, there may be a mobile app, installed on the customer’s phone, that propels heartbeat messages toward the SMP through the communal IP address. This is then actively programmed into the access control list of the home router or the edge router. This
technique may offer protected access to the smart bulb at the network level, along with a variety of IoT devices. A technique to improve the security of the Nest smoke alarm installed in smart homes and organizations can also be applied. This device is connected via a Wi-Fi network to cloud-based servers offering real time emergency alerts to the customer app. As the device holds the light and motion sensors, this becomes a valid anxiety that the device is able to track customers within the smart home and provide these details to Nest. There has been enhanced functionality that obstructs the Dropcam from uploading or recording video to the cloud when the customer is at home. It is amazing that this can be done automatically each time via the security alert. Therefore, certain suppliers provide IoT security as a service and energetically control the firewall procedures in support of the customer (with access switch of ISP or at the home gateway) that observes and controls the network level functions in support of every IoT device. Usually, by assessing IoT devices several at a time, all-purpose procedures can be implemented for all IoT devices, so that the improved security assertions can then be made available to the device manufacturer.

4.4. Prevailing standards and initiatives

With the increase in Internet access, the adoption of IoT in smart homes is also increasing. Correspondingly, IoT security concerns are also increasing. The market size of European smart home security is estimated to reach 7.95 billion USD by 2024 (Sovacool et al. 2020). The smart home security marketplace can also be attributed to the growing crime rate around the world. Due to the growing crime rate, customers are gradually paying more attention to security and safety systems, particularly in the housing sector. In order to combat the challenges, a number of home security standards have been developed to offer interoperability among the large range of associated products and services (Khoa et al. 2020; Sovacool et al. 2020; Sowah et al. 2020). The most important organizations involved in the development of the significant standards include:

**Open Connectivity Foundation (OCF):** In the smart home space, OCF became the biggest traverse business association in 2016, and its contribution in the amalgamation of the two most important proposals in support of interoperability is really remarkable. AllJoyn and IoTivity are the open source frameworks, which have been sponsored by OCF. Both of these standards work towards communication and encourage interoperability in technologies within the IoT. The recent creature is also functioning on an integrated IoT standard sustaining OCF stipulation.
Zigbee Alliance: Fundamentally amalgamated in smart home systems, Zigbee technology is implemented through the leading global service installers, suppliers and dealers. Supported by more than 400 associates, the ZigBee home automation standard is devoted to interoperability between an array of products irrespective of the producer. In 2017, the CES alliance completed a considerable move to advance interoperability with the creation of Dotdot, which is predicted to be a worldwide language in support of IoT. Dotdot is derived from the improvement of the widespread applicative upper level, which allows interconnection between Zigbee products and other products using Bluetooth and WiFi protocols. The ZigBee Alliance is approaching the standard in support of home devices, from lighting systems and temperature control to smoke detectors and security monitors. The ZigBee requirement is well-matched to smart home systems due to its consistency, low power consumption, and interoperability.

Bluetooth Low Energy: Marketed as Bluetooth Smart, Bluetooth Low Energy has been promoted by the Bluetooth Special Interest Group (SIG) as a smart, power friendly adaptation of wireless technology, in support of dot to dot contacts. Bluetooth provides the framework for direct connection from a tablet or smartphone, enabling customers to manage their smart home devices from a portable device. The Bluetooth SIG suggests that the Bluetooth mesh protocol should be favored. A mesh protocol would enhance the physical variety of devices via the Bluetooth network and it may possibly drop the power utilization.

IFTTT (“If This Then That”): This is a web tool that enables numerous applications and services to be linked using commands that trigger and automate actions. Within the smart home region (Sovacool et al. 2020), IFTTT enables devices to communicate with each other openly, without performing as a hub to exclude a channel. For example, if thermostat X’s temperature reaches 90°, the system Y must activate an alarm. IFTTT is implemented and utilized through several considerable performers within the smart home. IFTTT perform operations through Samsung’s SmartThings, LIFXlight bulbs, Nest thermostats or within Google Home.

Open APIs: Several industries have improved platforms to create devices that work together using Samsung’s SmartThings or Apple’s HomeKit. Accordingly, developers of related inventions are able to execute SmartThings as well as Homekit using similar software, or in a few cases, with hardware growth. The most recent competitor is Amazon with its smart voice sponsor Alexa within the Echo device, which provides an innovative and easy method to interrelate products. Amazon has unconfined the smart home skill API with the Alexa Skills Kit (ASK), which allows innovators to provide Alexa with skills or abilities, allowing Alexa to communicate
with the related inventions. Eventually, these platforms could possibly be converted into de facto standards, the same as the system device. The creator would profit from the improved visibility while customers would be able to attach and utilize them without difficulty (see: https://ec.europa.eu/growth/tools-databases/dem/).

**Thread:** Thread is an innovative low power wireless network standard for smart home security. The standard protocol sustains IPv6 by 6LoWPAN (Sovacool *et al.* 2020). The proposal behind Thread is the resolution of consistency, safety, power, and similarity concerns that regularly occur within smart home products. Thread is put on the physical layer that becomes the source of ZigBee devices, where an OEM is able to simply renew its ZigBee devices to sustain Thread through the development of software. Thread is very useful and provides an extra inclusive solution than earlier low-power wireless standards.

**Constrained Application Protocol (COAP):** This is a standard communication protocol for resource restrained devices in IoT. Several IoT deployments need proxies for asynchronous communication involving edge devices. This actually enables the proxies to access the vulnerable parts of CoAP messages (Gunnarsson *et al.* 2021). The latest standard protocol Object Security for Constrained Restful Environments (OSCORE) offers back-to-back security in support of CoAP messages by third party proxies. It executes the predictable services, through the improvements of significant security and privacy. OSCORE resourcefully offers sensitive trustworthy security and encryption on the dissimilar parts of CoAP messages (Gunnarsson *et al.* 2021). To evaluate whether the related security aspects consume a large amount of the limited resources that exist on a restricted device, OSCORE is implemented as an open-source protocol and its effectiveness is evaluated. OSCORE has been standardized to a large extent under the Internet Engineering Task Force (IETF) (Selander *et al.* 2019).

**Datagram Transport Layer Security (DTLS):** This is an Internet standard based security channel providing the transport layer that secures communications in excess of untrustworthy datagram protocols (e.g. UDP). Security is ensured by using two nodes that become the adjoining transport layer hops. DTLS is a secure reproduction of the TLS protocol (Dierks *et al.* 2008) with corresponding security assurances. For example, it prevents eavesdropping, interfering, tampering and message imitation. DTLS, in particular, is modified to support the use of UDP rather than the transmission control protocol (TCP). The innovative CoAP requirement (Shelby 2014) specified DTLS as a single security method to secure the CoAP exchange messages. There are two communicating devices, which primarily use the DTLS handshake protocol to swap cryptographic key material and network information to support message security. Specifically, one device performs like a
client, while another performs like a server. The defaulting handshake relies on documentation, excluding annexes derived from symmetric pre-shared keys (Eronen 2005), or on raw public keys (Wouters et al. 2014). A comprehensive handshake determines a protected session, wherein the client and server are able to start swapping data secured by considered key material. Secure communication is subsequently accomplished through the use of the DTLS record protocol, which ensures the consistency and security of message transfers.

4.5. Conclusion

The IoT is not a perfect application in a smart home as the security is always very critical. Even though the overall nature of security threats is also related to other domains such as confidentiality, authentication, and access threats, the vulnerability is still high considering the sensitivity of home-based applications. In addition, because the networked system is user-friendly, the physical accessibility of the systems leads to another big threat. The heterogeneity of the different devices and the fixed firmware further contributes to the system complexity. A number of solutions have been proposed that recognize and block the related threats at the network level and can contribute to the security of IoT-based smart homes. However, there is still a big gap to be bridged to offer seamless network security across all usable devices in IoT-based smart homes.

4.6. References


Drahansky, M., Paridah, M., Moradbak, A., Mohamed, A., Owolabi, F., Asniza, M. (2016). We are IntechOpen, the world’s leading publisher of Open Access books built by scientists, for scientists TOP 1%. Intech I(Tourism), 13.


5

IoT in a New Age of Unified and Zero-Trust Networks and Increased Privacy Protection

Sava ZxivanoVich¹, Branislav Todorovic², Jean Pierre Lorre³, Darko Trifunovic², Adrian Kotelba⁴, Ramin Sadre⁵ and Axel Legay⁵

¹ Technology Partnership, Belgrade, Serbia
² Institute for National and International Security (INIS), Belgrade, Serbia
³ Linagora Grand Sud Ouest, Toulouse, France
⁴ VTT Technical Research Centre of Finland Ltd, Espoo, Finland
⁵ UCLouvain, Ottignies-Louvain-la-Neuve, Belgium

In a new age of integration of many different services and zero-trust networks, the IoT has to overcome hardware limitations and provide adequate security. Our goal is to provide methods and effective tools to help build a network that is both easy to use and secure. It is based on Pi Platform for Unified Secure Communications, Services and Web-Applications, providing a foundation for home/small/medium office hubs. IoT security is based on the zero-trust philosophy – all communications are encrypted, no unidentifiable device is accepted on the network, each device has to provide its passport with a description of its network behavior and better encryption key generation. This IoT solution also addresses potential configuration problems by allowing trusted third-party providers to remotely access and correctly configure home/small/medium office hubs as well as informing users about any potentially harmful behavior or external access to their equipment.
5.1. Introduction

With the advancement of new technologies and the rapid proliferation of devices at home and in cities, users are now faced with a myriad of possible security vulnerabilities. The risk is particularly large, especially with IoT environments that increasingly integrate the devices of our daily lives at home and in the city, including electrical and gas appliances. In this context, any stolen private data can be used to manipulate sensitive information that endangers the personal, social and financial lives of citizens.

Privacy and data protection are fundamental human rights that are strictly established within the United Nations (UN). A prerequisite for this is building strong rules and technology frameworks to empower citizens and protect personal data and privacy. However, some members of the digital society are more vulnerable as they are less prepared to confront cyber-attacks and personal data breaches, such as biometric data breaches, resulting from a lost or stolen fingerprint and facial recognition data or through malicious mobile apps; or data breaches in the gaming industry, compromising million of accounts containing usernames, email addresses, IP addresses and hashed passwords. The scale, value and sensitivity of personal data in the cyberspace domain, in particular, the IoT domain, are significantly on the rise and citizens are typically uncertain about who monitors, accesses and modifies their personal data. Personal data breaches may facilitate abuse by third parties, including cyber-threats such as coercion, extortion and corruption.

One of the fundamental privacy challenges is the possibility that, in systems that depend on user settings, a large number of options may not be adopted by the general public due to the difficulty in their use. In addition to this, for the most part, service providers still rely on the traditional limited consent-based model, fostering binary (“allow/deny”) systems that do not easily allow the management of large quantities of data. On the other hand, full granularity of choice for each data set, authorised party and purpose may engender “consent fatigue” and alienate users. This reflects a tension between granularity and usability, both of which need to be taken into account during the design of the cloud and mobile services (Hansen and Limniotis 2018). From the point of view of security, the main challenges are related to the lack of adoption of client side encryption or layered encryption. On related security challenges, stronger authentication measures are needed, as well as more transparent procedures for dealing with data breaches and other incidents. In addition to this, due to the diversity of the IoT domain, it seems necessary to have different levels of security and privacy protection.

The history of the IoT also reflects its weaknesses, in relation to data privacy protection in particular. Increased data flow through all local networks on a daily basis renders every attempt by individuals, companies and institutions to follow and control it impossible, except for those who are highly specialized. Network traffic also usually
contains a certain amount of sensitive data, often digitally manipulated in some way (e.g. a variety of formats and data structures required by different applications). Such a situation makes a network breach or misuse of information even simpler during the course of an ordinary IoT operation.

There still a need for novel solutions to better protect users’ personal data, to ensure that data usage remains consistent with laws and legality and to help citizens to better monitor and audit their security, privacy and personal data protection, enabling them to become more engaged and active in the fight against cyber-risks. These innovative solutions can benefit from a self-sovereign identity, distributed ledger technologies and federated data processing to accomplish these needs.

This chapter outlines a first attempt to provide a solution to security and privacy challenges of the Internet of Things. The main contributions are as follows:

– To introduce the zero-trust philosophy and architecture for improved overall network security and better encryption.

– To define the use of Unified Secure Communications, Services and Web-Applications as a foundation for home/small/medium network hubs based on existing, tested and verified solutions (e.g. Pi Platform).

– To discuss methods and effective tools that could assist in creating networks that are both easy to use and secure not only for network members, but for trusted third-party providers and other remote users.

The next section presents a brief overview of the IoT with particular attention paid to security and data privacy. Section 5.3 highlights security and privacy challenges in order to clarify this issue and lay the foundation for the solution that will be proposed later in this chapter. In section 5.4 the authors have tried, within the chapter’s limits, to draw the outline of the current state of the art in the field of IoT security. Its purpose is twofold, to assist the reader in understanding the relation the proposed solution has with the current competitive solutions and to underline the main streams of the current work on the global level in order to emphasize differences with the new solution. Section 5.5 presents a zero-trust approach for security and privacy protection. This section provides all answers to problems listed previously, defines capabilities for applications, including compatibility and interoperability, and sets grounds for a practical example. In combination with the specific use case, section 5.6 demonstrates the applicability of this zero-trust approach in the form of a secure and private interactive intelligent conversational system and paves the way to the final sections on discussion and conclusions.

5.2. Internet of Things

The term Internet of Things (IoT for short) is neither precisely defined nor is its history exactly known. It is likely that it was first used in a presentation by Kevin
Ashton in 1999 who, at the time, worked for Procter and Gamble (Ashton 2009). In that presentation he presented his reflections on using new technologies such as RFID to track a large number of physical objects and collect information from them without human involvement.

Today, the IoT has become an umbrella term for technologies enabling the communication, acquisition and processing of information from various kinds of networked devices. There is however no strong agreement on what types of devices and communication exactly belong to the IoT. Some authors, especially those in the tradition of Wireless Sensor Networks, consider only small resource-constrained devices that use low-power and/or long-range communication to send sensor data to central data collection and processing points to be IoT devices. Others extend the definition to all sorts of devices with specific purposes that are deployed on a large scale, such as home routers, smart home equipment and mobile phones.

The development of the concept of the IoT has been enabled by progress in several fields in recent decades. The field of Wireless Sensor Networks has driven the mass production of low-power embedded systems with radio interfaces. The work on Wireless Adhoc Networks has contributed new network protocols that allow an efficient and robust exchange of information in unreliable networks. Thanks to Cloud computing, the large amount of data produced by an Internet of Things deployment can be processed in an adaptive and flexible way without requiring its users to acquire and manage their own server infrastructure. Finally, Machine-to-Machine application protocols, such as CoAP (Shelby et al. 2014) or MQTT\(^1\), are used to exchange information between the various components without the need for human intervention.

Zhao and Ge (2013) introduced a widely accepted general model of the IoT with three layers which we present here in a slightly extended version that allows for IoT devices with actuator capabilities:

– The **perception layer** consists of the IoT devices, i.e. field devices with sensor or actuator functions. For cost reasons and convenience (changing batteries), these devices are often resource-constrained in terms of computation, memory and energy and are based on embedded hardware platforms, running small optimized operating systems such as RIOT\(^2\) or Contiki\(^3\).

– The **network layer** is responsible for the collection, aggregation and transmission of the data sent or received by the devices in the perception layer. In modern IoT deployments, this layer not only comprises the communication networks and their protocols, but also the necessary infrastructure (often in the cloud) for storing and processing the data.

---

– The application layer contains all the applications and solutions driven by the sensory data. Various free and commercial frameworks exist to support the design and implementation of IoT applications. The larger vendors, such as Amazon⁴ and Google⁵, also provide management, storage and data processing services located in the network layer. Some vendors offer solutions that target specific use cases, most notably the smart home market, for example Samsung’s SmartThings platform⁶, and industrial applications⁷, the latter called the Industrial IoT.

The fulfillment of this model required a small embedded operating system (OS) that would be capable of enabling data transfer from IoT devices to the Internet. Due to a variety of IoT applications with specific functions and inputs/outputs, there are quite a few IoT operating systems in use to cover different requirements and demands. Almost every manufacturer and developer has their own preferences and ranking of IoT OSs. Therefore the list of IoT OSs in Table 5.1 is just an indication with regard to variety. To the best of our knowledge not one of the listed IoT platforms provides complete transparency on what data can be collected or means for users to select and define which data can be processed. The documentation also does not specify ways to recover from a cyber attack nor tools to protect hidden data that could be exploited by third parties. It was felt that more responsibility is put on the developers who use these solutions. However, there is an increasing awareness among platform vendors of the necessity of providing online management services, performing regular OS patch updates and scrutinizing all incoming and outgoing network traffic.

Furthermore, the significant growth in IoT deployment has led to the emergence of IoT platforms that provide users with the ability to quickly build, test, deploy and iterate on IoT-specific applications, supporting (Hammi 2018):

1) Easy integration of new devices and services.
2) Communication between devices (objects and servers).
3) The management of different devices and communication protocols.
4) The transmission of data flows and the creation of new applications.
5) Interoperability among components, objects, gateways, cloud data and software applications.
6) Scalability of the IoT infrastructure.

⁴ https://aws.amazon.com/de/iot/.
⁵ https://cloud.google.com/solutions/iot.
According to the level of services provided, IoT platforms can be divided into (Hammi 2018):

– Infrastructure-as-a-service backends: They provide a hosting space and processing power for applications and services, e.g. IBM Bluemix\(^8\).

– M2M connectivity platforms: They focus on only the connectivity of IoT objects through telecommunication networks and protocols, e.g. Comarch\(^9\) and AirVantage\(^10\).

– Hardware-specific software platforms: Numerous companies sell their proprietary technology which includes the hardware and the software backend, e.g Google Nest\(^11\).

– Enterprise software extensions: Some software and operating system companies, such as Windows and Apple, increasingly allow the integration of IoT devices such as smartphones, connected watches and home devices.

### 5.3. IoT security and privacy challenges

In this section, we will present challenges in IoT security that have been identified by experts, followed by an analysis of the risk situation related to the manipulation and treatment of privacy-sensitive data in IoT environments.

---


\(^10\) [https://airvantage.net](https://airvantage.net).

\(^11\) [https://nest.com/](https://nest.com/).
5.3.1. Security challenges

The increasing growth of the Internet of Things, either by putting new devices on the market or by developing new applications based on those devices, has raised many questions about the secure use of IoT platforms and associated services as well as the protection of users’ privacy. Sadique et al. (2018) discussed the current state of security in IoT and its challenges and identified issues such as device identity, firmware updating and installation if new patches are available, implementation of security algorithms knowing that IoT devices are often resource-constrained, with limited power, and finally, trust between different components in the IoT paradigm. However, IoT security is not only limited to the IoT devices themselves. For binti Mohamad Noor and Hassan (2019), the objective of IoT security is privacy protection, confidentiality and the security of the users, infrastructures, data and devices, as well as the guarantee of the availability of services offered by an IoT ecosystem. In this context, Ogonji et al. (2020) propose a privacy and security taxonomy which highlights threats, attack surfaces, vulnerabilities and countermeasures.

That the concerns of experts about the security of the IoT are not unfounded has been demonstrated in many recent incidents of security exploits and data breaches, such as the hacking of smart home devices (Marotti 2019) or home security cameras (Vigdor 2019). Systematic studies by researchers have revealed numerous vulnerabilities in products and services targeting the smart home market (Fernandes et al. 2016; Kaflé et al. 2019). Even if an IoT device and its communication are secured, the complexity of the involved interactions among participating entities (devices, cloud services, mobile apps) results in security hazards that significantly increase the size of the attack surface (Zhou et al. 2019). Sometimes, the hazard is not accidental and/or is exploited by the device manufacturer or the service provider themselves (McGregor 2019; Osborne 2019; Thapliyal 2019).

To date, the issues identified are still topical, as no consensus has yet been reached to regulate this area. Attempts to arrive at standards are still ongoing. It should be noted that there are several links in the IoT chain, such as communication protocols, data formats, etc., which require hard exchanges among many actors. While some authors (e.g. in Hassan et al. 2020) argue that new developments such as the Narrowband-Internet of Things (NB-IoT) improve privacy and reliability of transmission, it seems that existing standards for secure communication are not sufficient. When comparing security architectures, Ammar et al. (2018) show that different methodologies are followed by individual actors to provide other security properties in addition to the standards used for securing communications.

12. NB-IoT is a standard-based low-power wide-area network (LPWAN) technology developed to connect a wide range of new IoT devices and services.
The observations above are also valid for the so-called Industrial Internet of Things (IIoT). According to Liberg et al. (2020), the IIoT is part of critical system operation in many industrial use cases. Consequences of faulty operation can be disastrous. This is even truer when considering the pharma industry where IoT sensors and detectors can be used to supervise different styles of biomaterials and chemicals, detect tool flaws, and can be also used to assist in the prevention of fraudulent drug activities (Deepak et al. 2020). Munirathinam (Munirathinam 2020) considers security and privacy to be the number one challenge faced by the Industrial IoT.

As we have said, due to the diversity of the IoT domain, it seems necessary to have different levels of security and privacy protection. The control of a street lamp does not have the same security and privacy requirements as a portable health gadget. Once identified, these levels can be used to automate and personalize security and privacy services. The rapid evolution of the IoT domain and the emergence of new IoT products of all kinds lead the way towards reuse and adaptation strategies for security and privacy in order to avoid compromising platforms that are already operational.

Before continuing, it is necessary to clarify some very important concepts used in the rest of this chapter.

**Definition 5.1.** — Cyber security risk is the possibility of losing data or/to take control of your system, letting your private data be exposed or cyber-bulling, leading to financial or physical harm.

**Definition 5.2.** — Cyber-bullying\(^{13}\) is often based on altered information, helping to propagate false ideals and news. Cyber-bullying is defined as using electronic technologies in order to bully another person through the Internet. Unfortunately, cyber-bullying very often stays unreported.

### 5.3.2. Privacy challenges

In the past, user data were limited to a few pieces of profile information, and the risk of exploiting this information was minimal. Today, the risk is great because more and more systems are tracking and storing even the smallest amount of user usage data, known as data analytics. Advances in artificial intelligence and communications networks such as 4G and 5G have also expanded the scope of data that can be exploited to include voice, video, and image. As a result, the user has lost any possibility of preserving their privacy. With the increase in attacks against systems for data theft and malicious use, prices for security services will rise sharply. The result is that normal users and small businesses will be the most exposed and affected by the manipulation of their private data.

---

Practical applications of solutions that handle privacy issues in the EU in general follow the guidance of corresponding institutions on the European level. The latest corresponding initiatives in the EU are related to the European Self-Sovereign Identity Framework (ESSIF), as part of the European Blockchain Service Infrastructure (EBSI). The EBSI is a joint initiative from the European Commission and the European Blockchain Partnership (EBP). EBSI provides a common, shared and open infrastructure based on blockchain technologies aimed at providing a secure and interoperable ecosystem that will enable the development of EU-wide cross-border digital services in the public sector. The driving concept of the EBSI, according to the EBP mission and vision, is to

enhance efficiency, security, transparency and engagement, providing an interoperable framework for data and services that from one side enables the key EU visions (Once Only, Single Digital Gateway, ...) and at the same time allows each participating entity to run cross-border or internal services with secure access to needed information while maintaining autonomy running its own processes with its own technology stacks, regardless of the processes and technologies of any other entity.\textsuperscript{14}

In this sense, the implementation of decentralized biometric credential storage options in the EU is often proposed via blockchains using DIDs and DID documents within the IEEE 2410-2017 Biometric Open Protocol Standard (BOPS). Decentralized identifiers (DID) are a type of identifier that enables a verifiable, decentralized digital identity. They are an important component of decentralized web applications.\textsuperscript{15}

Also worth mentioning is eIDAS (electronic IDentification, Authentication and trust Services), EU regulation no. 910/2014 on electronic identification and trust services for electronic transactions in the internal market. Adopted on 23 July 2014, it provides a predictable regulatory environment to enable secure and seamless electronic interactions between businesses, citizens and public authorities. From September 29, 2018\textsuperscript{16}, all organizations delivering public digital services in an EU member state must recognize electronic identification from all EU member states.

When developing tools for IoT, data subjects may have or may not have particular rights depending on the legal basis that the data controller uses. Specifically, as you can see for the right to be forgotten (the right to deletion), the data subject may not have the right of erasure if the legal basis used by the data controller is a legal obligation (e.g. data relating to criminal proceedings) or public interest (e.g. information processed by a public authority in line with a public task). Figure

\textsuperscript{14} \url{https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITALEBSI/Mapping+of+Vision%2C+Mission%2C+and+Goals.}

\textsuperscript{15} \url{https://en.wikipedia.org/wiki/Decentralized_identifiers.}

\textsuperscript{16} \url{https://ec.europa.eu/futurium/en/content/eidas-regulation-regulation-eu-ndeg9102014.}
5.1 is a very useful table which summarizes data subjects’ rights as conceived by the French Data Protection Authority.

<table>
<thead>
<tr>
<th>Right of access to data</th>
<th>Right to rectification of errors</th>
<th>Right to deletion/right to be forgotten</th>
<th>Right to restrict processing</th>
<th>Right to data portability</th>
<th>Right to object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consent</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓ Withdrawal of consent</td>
</tr>
<tr>
<td>Contract</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Legitimate interest</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Legal obligation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Public interest</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Protection of vital interests</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Figure 5.1. Data subjects’ rights**

Particularly, the following specifications should be considered when dealing with data subjects’ rights.

– The data controller may have other procedures in place for enabling a data subject to request the deletion of their data (e.g. by filling in an online form or sending an email to a particular email address) and might not wish to comply with the search and destroy program. In that case, there should be a way to indicate to the data subject the way to request the erasure, without necessarily having to notify the data protection authorities.

– Locating and erasing personal data is also processing, thus this will be processing personal data and will need a legal basis, which most likely will be consent or a contract. This will also need an additional legal basis for processing sensitive personal data (when applicable), which most likely would be explicit consent.

– When trawling publicly available registries and databases, some datasets may be protected under trade secrets and other confidentiality rules. However, inferred or derived data which occurred after processing the original data lose their protection because it will be easy to find subject’s initially provided data, e.g. sexual orientation based on postal code.

– Notifying the data protection authority should occur only if the data subject has exhausted all other means of trying to exercise their right to deletion (right to be forgotten) for instance, by contacting the data controller and the relevant deadline for action has passed, otherwise the authorities may dismiss the claim as invalid.
This extensive introduction to data privacy takes the formal form defined below:

**DEFINITION 5.3.** – **Privacy protection** is about control of access to personal or Small and Midsize Business (SMBs) data that is, by law, private or that is assumed to be private by individuals. Privacy protection is about protection of personal information that is a term that may be used in a slightly different manner by different people. In this document, personal information denotes privacy sensitive information that includes the following:

– Personal data is consistent with Article 4 of General Data Protection Regulation (GDPR)\(^\text{17}\): Any information relating to an identified or identifiable natural person. An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person.

– Sensitive data are in line with Article 9 of GDPR\(^\text{18}\) personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs or trade union membership, and the processing of genetic data or biometric data for the purpose of uniquely identifying a natural person or data concerning health or data concerning a natural person’s sex life or sexual orientation.

– Usage data: Data collected from computer devices such as printers; behavioral information such as viewing habits for digital content, users’ recently visited websites or product usage history.

– Unique device identities: Other types of information that might be uniquely traceable to a user device, e.g. IP addresses, Radio Frequency Identity (RFID) tags, unique hardware ID ties.

**5.4. Literature review**

The aforementioned privacy and security challenges are typically addressed by technical mechanisms, most prominently tools known as privacy-enhancing technologies (PET), e.g. encryption, protocols for anonymous communications, attribute-based credentials, trusted computing, multi-party computation, homomorphic encryption, data obfuscation, privacy infomediaries, and private search of databases (Mowbray and Pearson 2012; Encinas *et al.* 2015; Guasconi *et al.* 2018). The effectiveness of those privacy-enhancing technologies has already been demonstrated by researchers and in pilot implementations. However, apart from a few exceptions, for example, encryption for data in transit and in rest became widely

---

17. https://gdpr-info.eu/art-4-gdpr/.
used, privacy-enhancing technologies have not become a widely-used component in system design according to the European Union Agency for Network and Information Security (ENISA 2018a, 2018b).

Starting with IoT security issues in general, it can be noted as the general observation that

Today’s IoT devices are insecure and incapable of defending themselves. This is due to mainly the constrained resources in IoT devices, immature standards, and the absence of secure hardware and software design, development, and deployment. The efforts of defining a robust global mechanism for securing the IoT layers are also being hampered due to diversity of resources in IoT (Khan and Salah 2018).

A number of authors analyze various aspects of IoT security, e.g.

The Internet of Things (IoT) envisions pervasive, connected, and smart nodes interacting autonomously while offering all sorts of services. Wide distribution, openness and relatively high processing power of IoT objects made them an ideal target for cyber attacks. Moreover, as many of IoT nodes are collecting and processing private information, they are becoming a goldmine of data for malicious actors. Therefore, security and specifically the ability to detect compromised nodes, together with collecting and preserving evidences of an attack or malicious activities emerge as a priority in successful deployment of IoT networks (Conti et al. 2018).

In this case authors also list the existing major security and forensics challenges within the IoT domain and briefly discuss some aspects of targeting identified challenges.

Wide application of the IoT heavily influences the complexity of the problem, in particular due to fast technological advance in the field. For that reason some authors restrict their research of trends and open issues in IoT security to recent years, in order to provide a valid overview of the current state of IoT security research, the relevant tools, IoT modelers and simulators. binti Mohamad Noor and Hassan (2019) explain that

the results of IoT failures can be severe, therefore, the study and research in security issues in the IoT is of extreme significance. The main objective of IoT security is to preserve privacy, confidentiality, ensure the security of the users, infrastructures, data, and devices of the IoT, and guarantee the availability of the services offered by an IoT ecosystem. Thus, research in IoT security has recently been gaining much momentum with the help of the available simulation tools, modelers, and computational and analysis platforms.
In addition to individual attempts in analyzing and publishing topics related to IoT security, there are conferences, webinars and similar activities that gather experts in the field in search for common solutions to the problem. One such example related to multidisciplinary approach emphasizes that “as Internet of Things (IoT) devices and systems become more tightly integrated with our society (e.g., smart city and smart nation) and the citizens (e.g., implantable and insertable medical IoT devices), the need to understand, manage and mitigate cybersecurity risks becomes more pronounced” (Choo et al. 2021). In their special issue of the journal, Choo et al. try to cover topics of problem classification, detection, analysis, privacy issues and protection. Despite their efforts, in the conclusion they state that a number of challenges still remain to be addressed. Nevertheless, we present below the abstract of one paper from the special issue that deals with similar topics as this chapter. In “Proof of X-repute blockchain consensus protocol for IoT systems”, researchers from the Harbin Institute of Technology in China, Shandong University of Science and Technology in China, Tencent Research Institute in China, Ch. Charan Singh University in India and King Saud University in Saudi Arabia designed a reputation-based consensus protocol for blockchain-enabled IoT systems (Choo et al. 2021).

One result of various initiatives and works, seriously motivated by the real need to find a practical solution to IoT security issues, is the “zero-trust”.

The Internet of Things (IoT) connects billions of devices to the Internet and the number is still increasing, which makes it very challenging to secure the applications, data, users, and devices in the complicated system. The zero-trust security has shown great potentials for IoTs which follows the “never trust, always verify” principle. In the past few years, the zero-trust security has attracted attentions from both industry and academics. The zero-trust holds the principle that every attempt to access the resources in IoT should be verified before granting the access (Li 2019).

Due to its importance, zero-trust philosophy and networks have also caught the attention of large companies and institutions up to the governmental level. The USA, as often taking leadership in topics about technology, have published some guidelines regarding Zero-Trust Architecture.

Zero-trust (ZT) is the term for an evolving set of cyber-security paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources. A zero-trust architecture (ZTA) uses zero-trust principles to plan industrial and enterprise infrastructure and workflows. Zero-trust assumes there is no implicit trust granted to assets or user accounts based solely on their physical or network location (i.e., local area networks versus the Internet) or based on asset ownership (enterprise or personally owned). Authentication and
authorization (both subject and device) are discrete functions performed before a session to an enterprise resource is established. Zero-trust is a response to enterprise network trends that include remote users, bring your own device (BYOD), and cloud-based assets that are not located within an enterprise-owned network boundary. Zero-trust focuses on protecting resources (assets, services, workflows, network accounts, etc.), not network segments, as the network location is no longer seen as the prime component to the security posture of the resource. This document contains an abstract definition of zero-trust architecture (ZTA) and gives general deployment models and use cases where zero-trust could improve an enterprise’s overall information technology security posture (Rose et al. 2020).

Previously in this chapter we mentioned that the innovative approach might include Self-Sovereign Identity (SSI), though with the original use, not relying on solutions which are already available.

In the age of increasing digital interactions and analysis of user data, the concept of Self-Sovereign Identities has gained a large amount of interest. It promises its users more control and a more user-centric experience that, in contrast to previous user-centric efforts, does not have to rely on any centralised entities. The concept of verifiable claims has been extended by the Identity Registry Model as well as the Claim Registry Model. These decentralized registries were enabled by blockchain technology and although not a necessity the storage can be decentralized too. This only leaves the claim-issuers and their position of trust as centralised entities in the system (Mühle et al. 2018).

Some SSI solutions are built upon blockchain technology as this already provides decentralized persistent data and consensus, but joining these two is not a must, as shown in some research. Both blockchain-based and other Self-Sovereign Identity solutions show to fulfill most of the evaluation criteria. The importance lies in the differences between solutions in both variants. Blockchain-based solutions definitely meet more properties on average than the others. The scheme by van Bokkem et al. (2019) shows that it is possible to create an SSI solution without blockchain technology.

5.5. Security and privacy protection with a zero-trust approach

A zero-trust approach aims to provide a framework for a network of home hubs, mobile phones, cloud services and edge devices with advanced encryption and blockchain. The reference architecture is based on Pi Platform. Its design is recommended for private individuals and SMEs. With intrusion detection and a notification system, it will provide herd notification like with meerkats and herd immunity that will significantly increase Internet security. It provides communication
infrastructure for users, services and IoT-enabled devices. The main objectives addressed by this approach are listed below:

– **Security and privacy management for a big and heterogeneous volume of data:** The nature and amount of data generated by users’ devices need new processing mechanisms in order to achieve better security and data privacy, for example, selecting better encryption mechanisms for more sensitive user data. We need security and privacy management methods that can be rapidly deployed and do not rely on instantaneous data access and availability.

– **Real-time and adaptive mechanisms for data protection:** Real-time monitoring is a big challenge and an important factor to detect and react to any data security and privacy violation. The goal is put in practice by handling priorities and adaptive security mechanisms that are able to manage high data volumes, such as those produced by IoT devices.

– **Inter-site network security and data privacy management:** User data are actually generated and available in different environments, such as at home when using a game console, mobile when connecting to their home manager, in the city when tracked by a security camera, at work when calling their meal provider, etc. Facilitating efficient collaboration between multiple secure platforms deployed in different places and in the cloud, gathering, protecting and securing data related to the same user will be a large challenge that needs to be investigated.

– **Intrusion detection and proactive defence strategies techniques and tools:** Novel intrusion detection mechanisms and defence strategies, mainly those based on artificial intelligence, should be included to produce a more robust platform against attacks.

As shown in Figure 5.2, the ZTA will base its approach on SSI and Pi Platform\(^\text{19}\).

SSI means that the user IoT device will directly manage its personal information and control access over it. With SSI, the power to control personal data resides in the user and not in any third party granting or tracking access to these credentials. SSI is based on credentials: users have verifiable credentials of different personal information (information certified by trusted issuers, for instance, a manufacturer could provide a credential of production date for an IoT device so anyone could trust that credential as it was directly provided by the manufacturer). Users will own several credentials for different personal information. They could limit and share just the required personal information for the required recipient. Users use the credentials in a privacy-preserving manner whenever and wherever they want. No prior agreement is needed, and no intermediaries are needed: devices will securely exchange personal information directly between them. The result is a more flexible, more secure and more private personal information management.

\(^{19}\) Developed by Tehnolosko Partnerstvo Doo Beograd (Stari Grad).
Pi Platform provides several services presented below:

1) **Verified by Pi-based service**: An efficient blockchain implementation that can be used to verify images and documents. For example, it helps to distinguish between altered images and new and original ones and provide needed accuracy. Such an approach allows easy object verification. In addition, Pi Platform provides communication infrastructure for users, services and IoT-enabled devices. All calls/messages/services are encrypted using TLS 1.2 + AES 128/256. The intention is to improve it using quantum RNG using QRNG (Figure 5.2, point 5) chipset like SKT IDQ S2Q000 that will enhance the security of the users’ data by using quantum encryption technology to generate random numbers and create unpredictable secure keys and provide the QRNG cloud service as infrastructure.

2) **Vault by Pi service**: Uses QR code (Figure 5.2, point 1 and 7) as a part procedure, making human interaction and decision mandatory. QR codes could be generated by the Vault by Pi service and signed using Verified by Pi, limiting the number of bits in the QR code and making easier to read from a computer screen. Using the Verified by Pi signing service, an application on a mobile phone that reads data, will be able to verify that the content provided in the QR code is created by a particular website and limit the possibility of hijacking the QR code. To help organisations boost transparency concerning data processing operations and enable
data subjects to receive essential information in order to make informed decisions reducing the time and effort needed, compared to the current practices generated QR codes, will contain information about: which organisation is accessing data, for what kind of purpose, the data to be accessed, for how long they would like to access data and for how long they would like to keep data but limited to the Vault by Pi setting controlled by the user.

3) **Trusted Third Party service**: Provides privacy as a service (Figure 5.2, point 3). PasS will be based on encrypted non-complete data requiring additional data to be provided to decrypt the data. PasS could be hardware as well as a software-based system. Such a system would provide trust by design. Accessing PasS will be allowed via the Vault by Pi application on a mobile phone. Providing PasS cloud-based and private cloud-based services will increase the speed of implementation as users will not need to purchase additional devices.

4) **Family Hubs service**: Allows all communication to go through Family Hubs by Pi (Figure 5.2, point 4), parents will be able to set up rules against cyber-bullying or introducing new rules of Internet engagement for their children without a need to monitor all communications.

5) **Analyse trustworthiness of data sources and risk assessment service**: Aims to enable enhanced security for the emerging IoT market. Security of IoT devices is very often overlooked in favour of the price of the equipment. Enhancing security for the IoT (Figure 5.2, point 6) is based on sharing better generated keys for secure communications as well as on AI analysing the traffic patterns inside networks, allowing faster responses to potential threats. Each IoT device will have its passport that will describe its credentials and traffic requirements. To expand potential threat analysis, we would use federate learning about all devices to be able to recognise trends and more subtle hacks.

6) **Audit interface service**: This is used by certified organisations to check edge devices (Figure 5.2, point 2), to provide secure configurations and help users as needed.

5.6. Case study: secure and private interactive intelligent conversational systems

Artificial intelligence (AI) has opened new horizons for IoT applications benefitting from the huge amount of data produced by IoT devices. However, software design has not yet evolved in order to propose new methods and measures that bring security and privacy to these new AI-based technologies. In this chapter, we are particularly interested in the user’s interaction in the intelligent environment. We will focus on interactive intelligent conversational systems because they concern a large part of futuristic IoT applications and products, such as home assistants, robots and driving assistants.

Interactive intelligent conversational systems leverage a number of heterogeneous components which are supposed to work seamlessly while maintaining a high level of
responsiveness, robustness, accountability and data integrity. The simplest user case, consisting of asking “what is the weather doing today” implies a pipeline consisting of a DSP-based microphone array processor, a phonetic processing unit, a speech to text module, a natural language processor, a SAAS (Software As A Service) client capable of gathering the weather information, a natural language generator, a text to speech engine, and a behavior and animation controller capable of synchronising the movement of the GUI avatar with the output speech. On top of that, one needs to account for the fact that the speech to text and text to speech module might be deployed on the cloud and subject to potential data losses as well as malevolent data compromising. All these heterogeneous modules might be written in different languages and operate on different architectures and hardware, on desktop computers, embedded appliances or cloud-based servers.

The paragraph above suggests an obvious first challenge: the heterogeneous nature of interactive dialog-based applications increases the complexity of deployment, testing and security assessment. The current chapter aims at answering these challenges by designing and proposing a set of relevant components and tools by means of security and data privacy according to the architecture presented in section 5.5. To that end the following specific goals are set: (i) Establish measures for secure conversational systems in IoT, assuring the privacy of data from user interaction to the accomplishment of user request, (ii) Establish privacy procedures for the exchanged data, (iii) Formally specify modules and their composition to guarantee the functioning of a compound system, and finally, (iv) Show, via a scenario, how it works.

We will use LinTO\textsuperscript{20} infrastructure as our primary field of study on interactive conversational systems for the case studies. LinTO is part of an industrial initiative to design a smart conversational assistant for companies. It includes a set of innovative features for individual or collective use. It is based on the French speech recognition engine in the advancement development phase at LINAGORA\textsuperscript{21}. Some use cases include: hands-free processes for better productivity, facility management using vocal control in the office and voice integration to end-users applications.

5.6.1. LinTO technical characteristics

LinTO is an open-source client-server system that enables the conception, deployment and maintenance of complete software and hardware clients that uses voice as a natural user interface. LinTO includes transcription services for simple commands and a large vocabulary. The standalone version is an embedded board with limited resources, a touch screen, speakers, a matrix of microphones and,

\textsuperscript{20}https://linto.ai/fr/.
\textsuperscript{21}https://www.linagora.com/fr/.
according to the configuration, a panoramic camera can be plugged into the board. It is complemented by a software platform for conversational assistance that supports intelligent interaction modalities.

More advanced deployment uses a functional LinTO platform stack running on a server and a LinTO client running on a device. The messages coming from the device are addressed to the platform using MQTT topics. The client could for example send an audio voice request to the server for further processing or stream an audio flux for live transcription. In contrast, server to client messages are used to control device behaviors such as speech synthesis and command execution. Currently, LinTO client is available for Raspberry Pi and Android. In addition, LinTO provides a user block programming interface where it is possible to create different applications through the combination of different components and services.

Given the potential usage permitted by LinTO, such as voice control of remote devices, it becomes important to secure the data flows, including privacy data such as user identities and locations. This is also taking into account the performance issues that may arise due to the complexity and size of the audio data. The zero-trust architecture presented above is used to provide that protection. A running example is presented in the following to show how this is being implemented.

5.6.2. Use case

Lisa is an elderly woman who lives alone but finds it difficult to move easily around her apartment to carry out even the simplest daily activities. An association offers her the secure LinTO device to control her most commonly used devices such as TV, heating, the front door and her medical bracelet. An example of the dialogues are shown below:

[case 1]
Lisa: LinTO
LinTO: Yes, ma’am
Lisa: Turn the heat up to 22 degrees
LinTO: Action executed, do you want something else?
Lisa: No
LinTO: (standby)

[case 2]
Lisa: LinTO
LinTO: Yes, ma’am
Lisa: Turn on the TV
LinTO: Action executed, do you want something else?
Lisa: Activate the series -Call the Midwife- on Netflix.
LinTO: Started in progress, please wait!
LinTO: Action executed, do you want something else?
Lisa: No
LinTO: (standby)

[case 3]
-Mobile device rings-
-Lisa picks up the device-.  
LinTO: Ma’am, your heartbeat is not regular.
Lisa: Contact the doctor
LinTO: Yes, ma’am
LinTO: Hello doctor, I’m the intelligent assistant,  
   madam has an irregular heartbeat.
Doctor: Send me all the data of last 3 days
LinTO: Data being transmitted, do you want something else?
Doctor: No
LinTO: (standby)

Figure 5.3. Zero-trust framework mapped on LinTO use case
5.6.3. Use case mapping on the reference architecture

Three types of components, presented by Figure 5.3, are used to develop the above scenarios: a mobile device where the LinTO conversational application is configured into command mode and is running, a home network gateway connecting all smart home devices to the network, and finally the client specific application providing a specific API (Application Programming Interface) that depends on the type of smart device functionalities. Each component is protected by the security and privacy features provided by the Pi Platform, detailed in section 5.5; this way all communications are protected from intrusion and personal injury.

The enhancement by Pi at all levels of IoT system deployment ensures consistency of inter-component exchanges and continuous verification at each entity of zero-trust architecture against any hacking attempt from outside parties.

5.7. Discussion

No system is really perfect, therefore the proposed IoT solution will also need to pass the tests of operability and usability, extended practical application, endurance to hacker attacks and various other misuses, and acceptance by the large number of users. Such proper testing in the field would require a period of commercial use, besides case studies and pilots. Furthermore, a complex solution like this IoT system requires well-organized handling of the feedback from users in order to perform debugging, fine-tuning and possible updates.

Some of the additional topics either directly connected to the proposed IoT solution or related to it in some way, might include:

– further enhancement of IoT data privacy, including accountability, verification of secure data origin and transparency of data;
– IoT secure information integrity;
– IoT emerging threats and risk management;
– IoT cybersecurity incident prevention, response and mitigation;
– physical security handling in IoT systems;
– enhancement of the system for IIoT (industrial Internet of Things) and Industry 4.0;
– IoT surveillance issues, including legal aspects.

The proposed IoT solution seems very promising so far and advanced encryption, SSI and block-chain have the power to overcome the limitations of the IoT with regard to data protection and privacy. At the same time, one must not forget that the very same advanced technology and block-chain have high computation requirements, restricted scalability, high bandwidth overhead and latency, making them overall
unsuitable to the concept of IoT in their basic form. The Pi-based service provides an efficient block-chain implementation, but the optimization of processes in order to run efficiently in the IoT might show some tradeoff of which we are not aware at the present stage.

Another aspect which cannot be predicted with certainty is the behavior and the acceptance of the proposed IoT solution by people. Besides the subjective attitude towards security (e.g. “it will not happen to me” behavior) and potential issues with SSI that in essence passes the management of personal information to the user’s IoT-device, there is also the problem of efficient standard testing and the presentation of results regarding the stability, performance, and security of the application. It is mandatory that a large number of people understand, accept and agree with the defined criteria. In this way, verification would be valid and users would have the motivation to use such block-chain based products.

5.8. Conclusion

Big enterprises and cloud service providers are collecting data about their users and limiting users’ options and limiting the functionality of systems that their users use. Such huge data gatherings are an enviable target and hackers are gathering data as they break into systems built by companies that are collecting data with users’ consent. A personal data breach may facilitate abuse by third parties, including cyber-threats such as coercion, extortion and corruption. According to the most recent IBM Security report, the average cost of a single data breach amounts to approximately 3.86 million dollars. Customers’ personally-identifiable information is the most frequently comprised type of record (80 per-cent of all breaches).

In this work, we recommend the use of the zero-trust approach at all layers of the IoT infrastructure. In particular, we recommend to perform a centralized configuration and management of devices, to access the health of IoT devices in a continuous manner, monitor IoT devices for anomalous behaviors, protect privileged identities, and slice networks into segments to minimize the impact of a potential intrusion. We also stress the need for automation and artificial intelligence-based methods to detect and quickly respond to a possibly ongoing attack.

The architecture presented in this chapter, advances the state of the art of usable security and privacy by proposing an application of self-sovereign identity (SSI) technology. With SSI, the power to control personal data resides with the user and not with any third party granting or tracking access to these credentials. Users can freely limit and share only the required personal information and for the required recipient. No prior agreement is needed and no intermediaries are needed: devices will securely exchange personal information directly between them. The result is more flexible, more secure, and more private personal information management.
5.9. Acknowledgements

This work was part of an initial project involving the following institutions and companies: VTT Technical Research Centre of Finland Ltd., Tehnolosko Partnerstvo Doo Beograd (Stari Grad), Huawei Technologies Oy (Finland) Co Ltd, Fundacion Tecnalia Research & Innovation, Hypertech (Chaipertek) Anonymos Viomichaniki Emporiki Etaireia Pliroforikis Kai Neon Technologia, Institut For Energiteknikk, Software Imagination & Vision Srl, Universite Catholique de Louvain, Vrije Universiteit Brussel, Linagora Grand Sud Ouest Sa, Institut Za Nacionalnu I Medjunarodnu Bezbednost, Halden Kommune, Centrul National De Raspuns La Incidente De Securitate Cibernetica, Nixu Oyj, City of Novi Sad.

5.10. References


6

IOT, Deep Learning and Cybersecurity in Smart Homes: A Survey

Mirna ATIEH\textsuperscript{1}, Omar MOHAMMAD\textsuperscript{2}, Ali SABRA\textsuperscript{3} and Nehme RMAYTI\textsuperscript{3}

\textsuperscript{1}Computer Science Department, Lebanese University, Beirut, Lebanon
\textsuperscript{2}Department of Computer Science, Lebanese International University, Bekaa, Lebanon
\textsuperscript{3}Computer Science Department, Varna Free University, Bulgaria

6.1. Introduction

Home automation implementation relies on a medium that supports communication and cooperation between devices. To achieve the desired objectives, it is necessary to adopt an approach that guides the implementation and design of such a home.

Smart home developers have solved many problems and addressed numerous issues to make the life easier through automation, or to help people (the elderly, children, people with disabilities) live a safe, independent life when necessary. However, many problems do not yet have solutions.

Cyber technology is now unavoidable in everyday life; note that wearable IoT devices, in relation to smart homes, have played a pioneering role in this technological revolution.

The more people rely on this technology, the greater the risk of infiltration and data leakage. Thus, the greater the need to develop a cybersecurity infrastructure that...
Cybersecurity in Smart Homes

protects these systems and their users, keeping in mind that it has now become clear that the IoT is vulnerable to many security breaches.

Additionally, IoT devices produce huge volumes and assortments of data, with varying degrees of veracity. Accordingly, when big data innovations are introduced, better and better information processing can be achieved.

Furthermore, as more devices become integrated into smart home solutions, the risk of attack increases, while securing smart home solutions becomes more challenging.

It is well known that IoT and smart home devices have low computing power, custom architectures and very little memory and storage while security solutions require a certain level of performance to operate. It is hard to port to custom architectures and requires a considerable amount of memory and storage for databases.

Cybersecurity experts have noticed an increasing trend towards machine learning-based solutions and most of them revolve around machine learning and deep learning techniques, especially if big data management is involved. This is because machine learning in cybersecurity looks for patterns in given data and requires very little computing power, memory and storage, it is easy to port to unknown architectures and it has the ability to send data to the cloud to analyze.

This chapter consists of seven parts. In the first part we present the problems related to the security of the various devices connected in smart homes. In the second part, we present the state of the art on smart homes and connected objects. The third part explains the IoT architecture and its different layers. IoT security is presented in part four and, in part five, we look at artificial intelligence, machine learning and deep learning, and the difference between them. We also define deep learning and the importance of its application to cyber security in Smart Homes. Part six is devoted to human activity recognition in smart homes, using neural networks and deep learning. To conclude, we present several methods for detecting anomalies and attacks in smart homes.

It is important to note that, when considering smart home network and device security for people with disabilities, we are dealing with a particularly sensitive topic in that it is related to human life and survival, and is not only about securing comfort and wellbeing.
6.2. Problems encountered

Gartner\(^1\) predicted that more than 20 billion IoT connected devices would exist by the end of 2020. These devices are not general purpose devices, such as smartphones and PCs, but dedicated function devices, such as retail vending machines, aircraft engines, smart cars, thermostats, wearable gadgets and a wide range of other examples (Hung 2017).

Gartner also stated that, by 2020, more than 25% of identified attacks in enterprises would involve the IoT, and yet the IoT would account for less than 10% of IT security budgets (Panetta 2016).

According to Kaspersky Labs, the number of malware samples for IoT devices has been increasing rapidly, from 3,219 samples in 2016 to 121,588 samples in 2018. It was made evident that there are a huge number of vulnerabilities within IoT devices (Kuzin et al. 2018).

In 2016, a distributed denial of service (DDoS) cyber attack was launched, causing major disruption to Internet services that affected some of the most technologically important companies, including Amazon and Twitter.

The cybercriminals behind the attack exploited the security weaknesses of thousands of IoT devices, allowing them to be hijacked and turned to be the sources of domain name system (DNS) requests that flooded traffic to the DNS hosting provider Dyn. It is worth bearing in mind that Dyn had DDoS countermeasures in place.

The DNS provider, Dyn, stores log data that has been efficiently processed by big data technologies and analyzed using deep learning algorithms, to determine any type of anomalous behavior. With over 20 billion connected things expected to be in use by the end of 2020, it is highly likely that this kind of DDoS attack is just the beginning (Gassais et al. 2020).

The increase in and spread of IoT devices has led to cyber security experts having to deal with new challenges, and it has widened the area of attack, starting with the smart home platforms themselves to the operating systems, communication media and the whole system in which it operates. This can lead to new types of attacks, such as denial-of-sleep attacks that drain the batteries of devices. These

\(^1\) Gartner is the world’s leading research and advisory company that equip business leaders with indispensable insights, advice and tools to achieve their mission-critical priorities. 
challenges are very real as many manufacturers of smart home appliances are solely focused on functionality; security is a much lower priority for them and some are not equipped to secure their devices against cyber threats. In addition, many IoT devices do not have the supporting infrastructure to run security solutions or even have updating mechanisms and, most dangerous of all is consumer negligence.

Furthermore, smart home solutions consist of tens or hundreds of IoT devices on the same network (in most cases these are wireless). Rather alarmingly, most of these devices have little or no protection at the software and infrastructure levels. The technology that was the science fiction of yesterday has become the reality of today yet, at the same time, it is making us more vulnerable to attacks. We do not wish to demonize these solutions, but we believe that smart home and IoT security must be taken more seriously.

On the other hand, smart home systems typically generate huge amounts of data from a wide range of sources and devices; these include sensors, situational data such as object locations, forecast data such as the weather, contextual data such as number of residents in the home, and operational data to manage the whole IoT system. These data must be converted into decisions and actions by suitable data science tools that are designed to work on big data.

Big data is high-volume, high-velocity and high-variety information that requires innovative forms of information processing for decision-making. Traditionally, big data is characterized by six basic characteristics, commonly known as the 6Vs. In general, data is classified as big data if it fulfills the first 3Vs: volume, velocity and variety. Big data technologies are the tools that are used to efficiently process big data.

Therefore, because the goal is to protect and secure substantial, high-value systems, and because the risks are high and multiple, and because huge amounts of data have to be handled by the network, the weekly or monthly security analytics reports would not be sufficient to detect and mitigate the cyber attacks in real time. Defensive tools that are efficient and advanced to the same degree as the systems and the attacks are needed.

In the recent past, most of us were used to having a laptop and a smartphone, each requiring the installation and maintenance of security solutions to protect them against attacks. Nowadays, some smart homes have more intelligent and interconnected devices than most medium-sized companies had some years ago.

It becomes difficult to handle updates, passwords, settings, etc. for each of these devices. For this reason, we consider the importance of adopting deep learning technology to secure each smart home solution, which in turn secures the whole system.

6.3. State of the art

Smart technology has been evolving for decades and, from time to time, a new concept appears. The smart home is one such concept and is the subject of several recent research works. Previous technologies include artificial intelligence, connected objects and cyber security. The concept of connected objects (Internet of Things, IoT) appears to have a promising future and will make life easier. In this chapter, we provide a clear idea of the IoT technology in smart homes by presenting the history, advantages and disadvantages, as well as the challenges of each.

6.3.1. IoT overview

IoT is a network of interconnected devices and tools with advanced capabilities to interact with other devices and also with humans and their environment to perform a set of tasks (Bari et al. 2013). To do this, we use sensors with a transparent connection between devices and the physical world. The new IoT devices have a wide range of sensors (microphones, light sensors, gas detectors, etc.) thus enabling more efficient applications (Lane et al. 2010). IoT devices can detect any change in their environment using sensors and take action to improve their operation (Suo et al. 2012); this has made it possible to make effective decisions. Communication between devices and the physical world has also made IoT devices operational in several fields of application (health, industry, household appliances, etc.). Indeed, the evolution of IoT tools has ensured the growth and development of the industry.

To guarantee the success of IoT technology, security should be guaranteed and vulnerabilities should be resolved. As mentioned previously, IoT consists of four layers; guaranteeing security at each of these layers means we can achieve complete security in IoT (Li et al. 2016). IoT depends on collecting information from physical objects and presenting them to the user through services and applications. Healthcare is an example of IoT technology that collects personal information about a patient’s health and transmits this to a healthcare system. Since a lot of personal
information is collected, this information should be protected from unauthorized users to maintain people’s privacy. The transportation of information should be protected from the sensor (source) to the application (destination) (Bertino 2016; Vyas et al. 2016).

IoT connects millions of devices in order to collect information. As the number of devices increases, the amount of information collected increases, thus privacy threats increase (Abomhara and Koien 2015).

To secure IoT, attacks should be prevented and security methods should be applied to prevent the vulnerabilities. Attackers will always target systems that have vulnerabilities, thus securing systems against attackers is the main goal because attackers are ever-present (Li et al. 2016) (Abomhara and Koien 2015).

6.3.2. History

IoT has been through several important development stages (Ibarra-Esquer et al. 2017):

– 1969: the Internet was born out of the ARPANET project;
– 1971: the first embedded systems\(^3\) appeared with the Intel 4004;
– early ’90s: the concept of ubiquitous computing was proposed by Mark Weiser;
– mid ’90s: the development of sensor nodes, wireless communication and digital electronics began;
– 1999: the term IoT was first used.

6.3.3. Literature review

Suo et al. (2012) refer to the challenges associated with the Internet of Things which stem from the following:

– IoT extends the “Internet” through traditional Internet, mobile networks, sensor networks and so on;

– every “thing” will be connected to this “Internet”;
– these “things” will communicate with each other.

Subsequently, Roman et al. (2013) focus on the distributed approach for IoT and the challenges related to the security of this architecture.

In 2014, an approach that describes challenges related to the security and privacy of IoT was presented. These challenges still need to be overcome in the coming years for maximum buy-in from all IoT stakeholders involved. Furthermore, a distributed capability-based access control mechanism was proposed, which is built on public key cryptography in order to cope with some of these challenges (Skarmeta et al. 2014).

Nobakht et al. (2016) focus on an intrusion detection and a mitigation framework called IoT-IDM to provide network-level security. They used machine learning to create patterns for some known network-level attacks and demonstrated this with a real IoT device: the “smart light bulb”.

Aly et al. (2019) present an extensive description of security threats and challenges across the different layers of the architecture of IoT systems. In addition, they focus on the solutions and countermeasures proposed in the literature to address these security issues.

Ahmad et al. (2019) focus on the modeling of the fog computing architecture and compare its performance to the traditional model. They present a comparative study with a traditional IoT architecture based on classifying applications, define a priority for each application, and use the cell operator as the main fog center to store data. Then, they give a solution to decrease data transmission time, reduce routing processes, increase response speed, reduce Internet usage and enhance the overall performance of IoT systems.

6.3.4. Advantages, disadvantages and challenges

Each piece of technology has its advantages, disadvantages and must overcome challenges in order to be usable, adaptable and secure for human use. The same goes for IoT (Yaakoub et al. 2019).
6.3.4.1. Advantages

The most important advantages of IOT are (Cognizant 2015; Sarmah et al. 2017; Soumyalalatha 2019):

– *communication*: IoT provides machine-to-machine (M2M) communication through which devices can stay connected, allowing full transparency and better performance and quality;

– *automation and control*: by using IoT, a huge amount of data can be automated and controlled by machines without the need for human intervention. This produces faster and more timely results;

– *saving time and money*: money and time are saved by monitoring different aspects of life using sensors;

– *new profit resource for businesses*: the sale of connected devices and related services exceeded $200 trillion in revenue through 2020. In addition, the value of IoT for organizations across industries is estimated to be $14 trillion in the next few years, which will likely lead to a 21% increase in global corporate profits by 2022;

– *improving productivity*: through JIT\(^4\) training and better labor efficiency;

– *improved quality of living*: IoT applications aim to make life easier and more comfortable;

– *new professions*: as new technological advances emerge, the opportunity for creating new jobs increases and therefore economic growth increases;

– *decision-making support*: vast amounts of information gathered by sensors and monitoring can aid better decision-making.

6.3.4.2. Disadvantages

Several disadvantages are noted in the literature (Arpita et al. 2015; Banafa 2017; Soumyalalatha 2019):

– *compatibility*: no international standardization of M2M protocols, variety of devices, firmware and operating systems used by IoT, non-consolidated cloud services;

\(4\) Just In Time inventory system.
– **complexity**: IoT architectures and networks are complex, thus hardware or software issues could lead to serious repercussions. Power failures may also cause disruption;

– **privacy**: an enormous amount of data is exchanged between devices and is monitored by various companies, making privacy breaches more likely;

– **unemployment**: humans are replaced by automated systems that are capable of performing many activities, which could result in increased unemployment rates;

– **controlling life**: the purpose of IoT is to automate activities and control various environments; as devices become more prevalent, humans will become more reliant on them;

– **possibility of malware spread**: the interconnectedness of devices imposes a risk of malware spreading throughout the home system, with consequences ranging from minor to very severe (Soumyalatha 2019).

### 6.3.4.3. Challenges

Despite the fact that connected objects do actually facilitate life, organizations face different challenges that represent barriers to growth. These challenges include (Cognizant 2015; Soumyalatha 2019):

– **scalability**: smart devices are connected to the network automatically, thus IoT should be capable of handling information management and service management issues across a wide range of environments;

– **self-configuration**: IoT devices should be automatically configured to be suitable for certain environments;

– **interoperability and lack of standards**: the various number of companies, technologies and protocols smart devices use, prevents interoperability where “connected systems should be able to talk the same language of protocols and encodings” (Zain et al. 2016). The lack of standards that allow smart devices to connect and communicate as desired makes it difficult for organizations to integrate applications and devices;

– **software complexity**: software infrastructure is required to support the network that smart devices connect to, since the latter operate with minimal resources;

– **storage**: smart devices collect enormous amounts of data that require scalable data storage to be allocated;

– **data interpretation**: context interpretation is important to generate useful information, and to draw conclusions from the data sent by the sensor;
– security and privacy: protecting data collected by smart devices from unauthorized use or attack is a major concern. Privacy concerns also arise from the massive amount of information supplied by users who are unaware that this information is being captured. Other challenges include hacking, criminal abuse and security breaches;

– energy optimization: a significant amount of energy is needed to operate several devices on the network, hence energy optimization is crucial to prevent shutdown;

– communication means: the networks used for connection and data exchange impose challenges such as availability, congestion and delays;

– data and information management: traditional infrastructure is not suitable for the sheer amount of data collected by smart devices, rather more advanced algorithms and systems are required to mine, analyze and derive value.

### 6.4. IoT architecture

The IoT architecture is composed of elements that fall into three categories:

– IoT hardware is composed of devices such as sensors;

– IoT middleware is composed of tools used for data storage and analytics;

– IoT presentation which is composed of tools used for data interception and visualization to keep track of various events occurring.

These elements are represented as four main layers which are: the sensing layer, the network layer, the service layer and the application–interface layer (Leloglu 2017) (Table 6.1.).

<table>
<thead>
<tr>
<th>Sensing layer</th>
<th>Radio frequency identification reader, sensors, gateway, GPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network layer</td>
<td>2G/3G communications network, Internet, mobile network, broad television network</td>
</tr>
<tr>
<td>Service layer</td>
<td>Information processing, cloud computing, data analytics, data storage</td>
</tr>
<tr>
<td>Application–interface layer</td>
<td>Medical applications, entreprise computing, transportation applications, mobile applications</td>
</tr>
</tbody>
</table>

**Table 6.1. Layers of IoT**
6.4.1. Sensing layer

The sensing layer is the layer between connected devices and the network layer. It collects information from the devices and passes this to the network layer. In the sensing layer, we have the IoT connected devices. Let us consider the four communication models for these IoT devices. First, there is the device-to-device communication model where devices communicate directly with one another using different types of networks. An example of this model is the smart home. Second, there is the device-to-cloud communication model where devices communicate with a cloud service using a wired or Wi-Fi connection. An example of this model is the smart TV. Third is the device-to-gateway communication model where devices communicate with the cloud service through the gateway. Finally, there is the back-end data sharing model where data is combined from different devices through multiple cloud services.

6.4.2. Network layer

The network layer is between the sensing layer and the service layer. It defines the communication between the connected devices to transfer collected information to the service layer to be processed.

6.4.3. Service layer

The service layer is between the network layer and the application–interface layer. It processes the information collected and saves it in the database for later use by the services that the user requires.

6.4.4. Application–interface layer

The application–interface layer is between the service layer and the users. It is developed based on user requirements or industry specifications.

At each layer there are multiple and various security risks that must be resolved. Therefore, each layer may have a different security solution for the information being collected from the devices before it reaches the application–interface layer, in order to achieve a totally secure system.
6.5. IoT security

Security is an important aspect of IoT to ensure the reliability, confidentiality and availability of the system. Imagine working with sensitive data on a system with weak security where anyone can intercept, read or modify this data. We treat security as a high priority when considering the development of the system. First, we consider the vulnerabilities and the risks found at each of the four layers in the IoT architecture. (Abomhara and Koien 2015).

6.5.1. Security in the sensing layer

The first layer is the sensing layer which gathers and exchanges information using sensors connected to the physical world (Li et al. 2014). The main concerns in this layer are:

– cost, energy consumption, and resources for the IoT devices such as sensors and RFID tags;
– deploying IoT devices at one time or at several times according to the requirements;
– using the hybrid network to connect IoT devices such as mobile networks and wireless networks;
– heterogeneity of IoT devices due to the huge amount of devices.

In this layer, security challenges and requirements are divided into two parts, one for the sensing layer and the other for the end devices connected in IoT. The security requirements for the sensing layer include device authentication, user authentication, confidentiality, integrity, non-repudiation, availability, access control, privacy and physical security protection. Device authentication is verifying the devices are trusted while user authentication is verifying user credentials in order to access the system. The security requirements for the end devices are data source authentication, device authentication, confidentiality, integrity, availability and timelessness. Examples of threats in this layer are: spoofing attacks, where the attacker masquerades as an IoT device and sends fake data; DoS attacks where the resource becomes unavailable to the users (Li et al. 2016).
6.5.2. Security in the network layer

The second layer is the network layer which describes how IoT devices are connected and specifies the medium used. The network layer includes communication technology like ZigBee (Ning et al. 2013) and 3G/4G/5G wireless communications (Ejaz et al. 2016). The main concerns in this layer are:

– QoS of data being transmitted;
– network management according to the type of network;
– confidentiality of information;
– privacy and security of user sensitive data.

The security requirements in this layer relate to overall security requirements, privacy leaks, communication security, fake network messages, overconnected devices, denial of service (DoS) and man-in-the-middle attacks. Many attacks can target the IoT communication protocols, such as eavesdropping against Bluetooth, NFC, Wi-Fi, etc. (Bapat et al. 2017). Eavesdropping and replay attacks are other common attacks in this layer (Vafaei 2014). Examples of threats in this layer include data breaches, where secure information is released to an untrusted environment, malicious code such as viruses, malware and Trojans.

6.5.3. Security in the service layer

The third layer is the service layer which stores data needed by the user for the applications. The main concerns in this layer are:

– the APIs of the service;
– management should be trustworthy;
– service discovery and composition to find the suitable service required by the user (Choi et al. 2012).

The security requirements in this layer are authorization, service authentication, group authentication, privacy protection and privacy leaks. Examples of threats in this layer are DoS attacks, unauthorized access to data, tampering with data, etc. (Atzori et al. 2010).
6.5.4. Security in the application–interface layer

The fourth layer is the application layer where various applications are presented to the users. The main concerns in this layer are:

– confidentiality, authentication, authorization and integrity between layers;
– safe software downloading, secure remote management and isolation of sensitive data (Gu et al. 2014).

The security requirements in this layer include sensitive information isolation, safe remote configuration, software downloading and updating, administrator authentication, a unified security platform, security patches, integrity and confidentiality for transmission between layers, cross-layer authentication and authorization, etc. Authorization is verifying the user has permission to access a resource. Examples of threats in this layer are: social engineering a very popular technique where an attacker obtains information from the users by tricking them, injection attacks, where an attacker executes the code on the server directly which causes data loss or data modification, distributed DoS, etc. (Ning et al. 2013).

This four-layer infrastructure maintains total security for IoT to avoid connecting fake devices, capturing data generated by attackers, as well as other risks. If we compare the risks at each layer, we find that the common risks are DoS attacks, lack of authentication, authorization, data confidentiality and privacy.

6.5.5. Cross-layer threats

Information exchanged between the layers may be vulnerable to threats such as:

– sensitive information being transmitted from one layer to another;
– leakage of sensitive information at the boundaries of the layers;
– misconfiguration.

The security requirements in this layer are security protection, privacy protection and trust (Li et al. 2016).

6.5.6. Security attacks

Since IoT is susceptible to many threats and vulnerabilities, attacks occur in different layers and typically concern four aspects of security: secrecy,
authentication, integrity and availability (Li and Da Xu 2017). Some examples of different attacks are presented below.

6.5.6.1. **DoS attack**

In this type of attack, an attacker tries to deny access to the resources and services which affect the availability of the services. DoS attacks involve requests and messages being sent to consume the resources of a certain device (Saadeh et al. 2016).

6.5.6.2. **Replay attack**

In this type of attack, an attacker intercepts the communication, copies the message, and sends it again impersonating the real sender. This leads to the theft or modification of messages being transmitted (Mahalle et al. 2013).

6.5.6.3. **Eavesdropping**

In this type of attack, an attacker only listens to the communication of data being transmitted between two parties but does not modify data, thus affecting the privacy of the data only (Saadeh et al. 2016).

6.5.6.4. **Physical attacks**

In this type of attack, the hardware components of IoT devices are attacked through configuration modifications (Abomhara and Køien 2015).

6.5.6.5. **Man-in-the-middle attack**

In this type of attack, an attacker takes the advantage of threats and vulnerabilities to intercept the communication of the data being transmitted between the two parties, to read or modify the data.

6.5.6.6. **Data modification**

In this type of attack, an attacker gains access to the data transmitted between parties so they can read or change the format of the data, affecting the confidentiality of the data received.

6.5.6.7. **Spoofing**

In this type of attack, an attacker sends a malicious tag to a sensor and this sensor treats it as a valid tag; this approach can result in full control over the system.
6.5.6.8. **Sniffing attack**

In this type of attack, an attacker can gain access to information by sniffing or monitoring network traffic using sniffing applications (Abdul-Ghani *et al.* 2018).

6.5.7. **Security requirements in IOT**

As fog computing is still in its infancy, there is a limited amount of work dedicated to certain security/privacy issues. While using the fog network at this stage, new problems and concerns will arise as a result of the properties of the fog computing network. For instance, fog network heterogeneity, the diversity of fog network framework, the need for low power connected end devices, and mobility holdup are some resulting problems of fog network. The research will focus on a systemic review of fog networks and platforms, determine the possible security gaps, analyze existing security solutions and list comprehensive solutions that can eliminate many potential security flaws within fog systems.

Since each layer has different features and roles, each layer also has different security requirements (Figure 6.1).

Figure 6.2 shows the six aspects of security requirements for the IoT framework which are: confidentiality, availability, privacy, authenticity, integrity and non-repudiation. Confidentiality is needed to secure the data; availability ensures that the resources and data are accessible; privacy is needed to protect customer data; authenticity components ensure proof of identity; integrity is to guarantee the data is accurate and trusted; and finally, non-repudiation is needed to provide a trusted audit trail. All of these security challenges make the data vulnerable and exposed to an attacker (Li *et al.* 2016; Pal *et al.* 2020). To ensure security across the system, we work on a solution in each layer.

![Figure 6.1. Security requirements in IoT](image)
6.5.8. Security solutions for IOT

Having discussed the security problems in IoT, we now consider recommended security solutions at each layer (Li et al. 2016).

6.5.8.1. Security solutions in sensing layer

To resolve the security problems and avoid attacks on IoT devices, four actions should be done:

– implement specific security standards for all IoT devices so all devices apply the same standards, in order to avoid loss of data in the event a device is replaced;
– build a trusted data sensing system to continue receiving data from trusted devices only;
– trace and identify the source of a user to ensure validity.

These actions can be applied in two steps at the sensing layer. The first step is to ensure that the users are authorized and authenticated before accessing sensitive data. The second step is to apply security methods for IoT devices to ensure the data is collected and transmitted securely. To ensure the privacy of the data, there are multiple techniques applied according to the type of the data. For example, encryption, decryption and hashing techniques are applied on text data while a CRC (cyclic redundancy check) and image compression techniques are applied on images. For RFID devices, multiple techniques are applied such as cryptography, AES (advanced encryption standard) and hashing algorithms. For the integration of RFID devices and WSN nodes, new challenges arise such as user authentication, communication security, privacy and others. These techniques are applied when the devices are manufactured. Basic security protocols for communication are authentication, availability, confidentiality and integrity (Li et al. 2014; Pal et al. 2020).

6.5.8.2. Security solutions at the network layer

To resolve these problems and avoid any attack on the network, two actions should be done:

– authentication and authorization to avoid any attack and validate the identities of the users;
– secure transport protocols to maintain secure transmission of the data through the network.
Security in this layer is divided into two sub-layers according to the medium: the wireless sub-layer and the wired sub-layer. Security solutions for the wireless sub-layer are concerned with developing protocols while security solutions for the wired sub-layer are concerned with securing connected devices (Li et al. 2016).

6.5.8.3. Security solution at the service layer

To resolve these problems and avoid service attacks, two actions should be done:

– secure transmission between the service and the other layers;

– secure service management to maintain secure service identification and secure access control.

In this layer, each service requires a particular method according to the features of the service. Examples of the methods applied are authentication, access control, privacy and information integrity (Choi et al. 2012).

6.5.8.4. Security solutions at the application–interface layer

Several actions should be taken on IoT devices to keep them secure and safe such as:

– design the cluster for the IoT devices based on an efficient energy solution;

– use lightweight security solutions for different IoT devices;

– focus on the safety of IoT devices.

In this layer, security solutions are applied according to the application being used. For example, local applications require encryption and steganography, while global applications (applications from an external network) require additional methods such as authorization, authentication, antivirus, as well as others. An example of an application is the SCADA (supervisory control and data acquisition) system which presents technical solutions to monitor processes in the industrial environment. In the SCADA system, some security methods are authentication, physical security, system recovery and backup.

The success of IoT depends on the guarantee of security at all layers of the infrastructure.
6.6. Artificial intelligence, machine learning and deep learning

Artificial intelligence (AI), machine learning (ML) and deep learning (DL) are three separate terms, however there is some crossover between them. Artificial intelligence encompasses both machine learning and deep learning. Machine learning, in turn, encompasses deep learning, as shown in Figure 6.2.

In the field of artificial intelligence, not all that is machine learning will be deep learning (Wasicek 2018; Oppermann 2019).

– *Artificial intelligence* is the science and engineering behind intelligent machines and programs.

– *Machine learning* means computers can learn without being explicitly programed.

– *Deep learning* is learning based on deep neural networks.

*Figure 6.2. AI, ML and DL. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip*
6.6.1. Artificial intelligence

Artificial intelligence (AI) is a concept that has theoretically been around for a hundred years. The first intelligent machines were developed in the 1950s. These work and react like humans. Artificial intelligence is any system, program or machine that can think, analyze, learn, make decisions and develop like a human. In other words, it is “the simulation of human intelligence processes by machines”\(^5\). Nowadays, AI applications include a vision and/or speech recognition system, an expert system, and natural language processing (NLP).

Intelligence programming is based on three essential elements: learning, reasoning and the process of self-correction (Burns et al. 2021):

– Learning means collecting data and formulating rules to turn data into actionable information.

– Reasoning is a cognitive skill that allows the appropriate algorithm to be chosen in order for the system to achieve a desired result.

– Self-correction means the algorithms of the system are continuously being developed to provide the best results.

6.6.2. Machine learning

Machine learning differs from traditional programming which requires manual coding. ML uses data to train the machine on how to perform a specific task. The input of machine learning is data and the output is a model (Figure 6.3).

![Diagram](image.png)

**Figure 6.3.** a) Traditional programming; b) machine learning. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip

---

Through machine learning, a system can perform a learning function with the data it ingests and thus it becomes progressively better. This learning is possible through the use of examples to improve some aspects of performance. The data is considered to be a set of training examples. The algorithms parse the training data, and use the individual training examples to see how well they can answer the question related to their goal. That answer is then analyzed and used to improve the algorithm’s ability to give better answers in the future.

This learning process is repeated for each example. In this way, each training example contributes a little bit to the accuracy or predictive power of the algorithm. If the learning process works, we say that the learning algorithm generalizes, meaning that its predictions are useful beyond the training examples (Wasicek 2018).

Like any other technology, machine learning excels at solving certain types of problems or tasks, whereas other technologies are more suitable for solving other problems. Below are three general problem settings that are well suited to a machine learning approach:

– **Classification**: sorting individual items into a set of classes, like recognizing anomalies in unusual sequences of credit card transactions.

– **Regression**: predicting outcomes based on historical records, like predicting the future of stock prices or currency exchange rates or which movies a person will like according to the historical records of the problem.

– **Clustering**: finding items similar to one another, like recognizing patterns in objects in real scenes, facial identities or facial expressions and/or spoken words.

Many machine learning techniques can be categorized into one of the four following types:

– **supervised learning** deals with labeled data and direct feedback. It can predict an outcome or future trends;

– **unsupervised learning** deals with unlabeled data and works without feedback. It is good at finding the hidden structures or patterns in data;

– **semi-supervised learning** falls in-between supervised and unsupervised learning and works well with partially labeled data;

– **reinforcement learning** focuses on decision processes and reward systems during progress. It can learn a series of actions.
6.6.3. Deep learning

Over the last few years, deep learning has become a prominent arrangement of machine learning techniques dependent on learning data representation. It has become apparent that deep learning algorithms can beat best-in-class approaches in conventional AI issues such as picture and sound classification (Dean et al. 2012). Furthermore, it has been expressed that they may outperform human-level abilities in classifying these sorts of information (Sparks et al. 2013).

Deep learning is a subsection of machine learning (and therefore of artificial intelligence) that is based on artificial neural network (ANN) models. The artificial neural network is an important programming paradigm. Previously, to make things easier for the computer, we split the big problems into small tasks. But, with neural networks, the computer has the ability to learn from training data in order to find the best solution. Since 2006, machine learning and deep neural networks have enabled deep learning. Deep learning provides better performance for certain problems like speech recognition and computer vision.

Neural networks are clustering and classification tools of data that we store and manage, according to their similarities or their labeling. In addition, artificial neural networks have the possibility to make a predictive analysis to establish correlations between items.

The word deep refers to the number of layers in neural networks. Deep learning currently plays a critical role in the development of highly automated systems.

The notable advancement of deep learning is the result of three fundamental elements:

– the collection of huge amounts of data;

– the turn of events and openness of new AI structures and machine learning platforms (Abadi et al. 2016, pp. 265–283) and algorithms (Niu et al. 2011) because of advances in equivalent (Raina et al. 2009) and adaptable programming frameworks (Gonzalez et al. 2012);

– storage costs have been quickly decreasing (Komorowski 2015) and mobile applications, IoT and the significance of information as an asset (Parkins 2017) have all prompted further interest in innovative work using deep learning technology (Press 2016).
Deep learning is based on a deep neural network. It is a subset of machine learning which is the ability to learn without being explicitly programmed, and it is a part of artificial intelligence, which is the engineering of intelligent machines and programs.

Deep learning involves learning data representations by utilizing a network of multiple layers of nonlinear preparing units for different sorts of highlight extraction and change. Each layer’s output is the successive layer’s input. Generally, deep learning models and techniques endeavor to copy the movement in layers of neurons in the neocortex, for example, an artificial neural network. It learns, like other machine learning strategies, by iteratively grouping a preparation informational collection, and refreshing its boundaries marginally into the correct bearing each time a characterization error happens. Eventually, the tweaked boundaries of the algorithm are tried on an assessment informational collection to quantify the algorithm’s performance. Such an algorithm works with the programmed classification of information which, when conveyed, eliminates the requirement for an individual to classify the information physically (Domingos 2012).

![Figure 6.4. Structure of the deep learning network](image)

### 6.6.4. Deep learning vs. machine learning

Deep learning has proved to be more successful than machine learning because deep neural networks (DNNs) have significant capacity for storing information.
(Figure 6.5.); on the other hand, DNNs do not need feature extraction which requires experts in the field of the problem to be solved.

**Figure 6.5.** Deep learning vs machine learning (Wasicek 2018). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip

Flat algorithms (in traditional machine learning) such as decision trees, SVM, or others require feature extraction before applying them to raw data. This step will be adapted and tested for a given task over several iterations to achieve the optimal result, using classic machine learning algorithms (Alabs 2019) (Figure 6.6).

**Figure 6.6.** Deep learning has no feature extraction step. For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip
6.7. Smart homes

A smart home can be characterized as a home equipped with sensors and a middleware framework, as well as communication interfaces that anticipate and react to the needs of the inhabitants to improve their comfort, enjoyment and security through the use of technology inside the home (Aldrish 2003). A smart home can support a variety of services and automated tasks from temperature control and smart climate control system to more complex requests such as monitoring the surroundings of an inhabitant or tracking their behavior or wellbeing in the home (Liu et al. 2016).

6.7.1. Human activity recognition in smart homes

Human activity recognition (HAR) is a unique and challenging research topic (Ranasinghe et al. 2016). The purpose of HAR is to determine activities performed by one inhabitant or numerous inhabitants depending on various sensors that are prearranged and configured to notice and detect many events, for example movement sensors, pressure identifiers, RFID tags, electrical power analyzers (Belley et al. 2015), and more. The HAR cycle includes a few stages. The four primary stages are as follows (Figure 6.7):

– **Pre-processing**: separating raw data from sensor streams to deal with inadequacy, remove noise and repetition, and perform data standardization and labeling.

– **Feature extraction**: separating features from raw data to utilize it as input to machine learning.

– **Feature selection**: reducing the number of features to improve their quality and lessen the computational effort required for classification.

– **Classification**: identifying the given activity using AI and machine learning.

The general objective of the HAR framework is to replace all – or almost all – of the human tasks inside the home, either by predicting these activities and acknowledging them when important or by meeting the necessities and requirements predefined by the residents. For instance, with the assistance of sensory devices, a HAR framework can monitor the medical issues of an inhabitant and alert healthcare services in the event of an emergency. (Rashidi and Mihailidis 2013; Zhao et al. 2019).
6.7.2. Neural network algorithm for human activity recognition

In the field of human activity recognition, neural networks have recently demonstrated a decent level of proficiency and precision in comparison to other machine learning algorithms.

The first application using ANNs in a smart home environment was developed by Mozer (1998). This application, named ACHE (Adaptive Control of Home Environments) was able to adapt the environmental conditions (heating, lighting, ventilation and water heating) to the needs of residents and their level of comfort (Kasabov 2002).

Jorge and Goncalves (2001) worked on automated monitoring of the health of the elderly using artificial intelligence tools. They collected data from the elderly regarding neurological disorders (loss of motor, sensory and cognitive abilities) via computer devices, with the aim of predicting the next activity (Elman 1990).

Pigot et al. (2003) tried to minimize the risks resulting from actions taken by elderly people in a physical environment, both at a theoretical and practical level. The authors applied ANNs with other mathematical models to aid in the detection of models associated with risk (Pigot et al. 2003; Stefanov et al. 2004).
Rivera-Illingworth et al. (2005) developed new connectionist architecture to recognize the behaviors of daily life (sleeping, eating, etc.), using simple sensors and an intelligent algorithm (Augusto and Nugent 2006; Montana and Davis 2006).

Three kinds of learning algorithms are required (Hiregoudar et al. 2014): supervised neural networks, unsupervised neural networks and semi-supervised neural networks. A brief description of each of the three calculations is given in Table 6.2. A typical neural network structure is shown in Figure 6.8.

ANNs can be categorized as either feed-forward networks or feedback networks, as shown in Figure 6.5. Each learning algorithm is intended for preparing a particular architecture. Consequently, when we examine a learning algorithm, a specific network architecture affiliation is implied (Jain et al. 1996). In Fang et al. (2014) back-propagation (BP) is utilized to prepare the feed-forward neural network for human activity recognition. This algorithm was compared with another probabilistic algorithm: the Naïve Bayes (NB) classifier and Hidden Markov Model (HMM). The outcomes show that neural networks that make use of BP calculation generally have better human movement acknowledgment exhibitions than the NB classifier and the HMM.

In Mehr et al. (2016), quick propagation (QP), the Levenberg–Marquardt (LM) algorithm and batch back propagation (BBP) have been used for human action recognition and compared alongside execution on the Massachusetts Institute of Technology (MIT) smart home dataset. The accomplished outcomes showed that the LM algorithm has better human activity recognition execution (92.81% accuracy) than QP and BBP algorithms. This is performed as if for a single occupant home. In the event of various clients, more complex learning is needed, with feature selection and more refined sensors.

Lee et al. (2017) proposed a technique based on a one-dimensional (1D) convolutional neural network (CNN) to detect and follow the activity of the person living in the house. They used the data collected (walking, running and resting) of the triaxial accelerometer from smart mobile phones. The speed of the activity has three parameters x, y and z which are transformed into vector magnitude information. These contribute to the learning of the 1D CNN. This technique had 92.71% accuracy. The accuracy of a neural network is based on the nature of the different characteristics, supervised exercises and limitations.

Hussein et al. (2014) created a system that gives people with disabilities the option to control aspects of everyday life or allow the system to automatically provide what is necessary for them to live independently, without the help of others. Their system is designed to monitor the elderly and people with disabilities so as to bring them more security and safety without disrupting their lives. Their behavior and living habits are
recorded using a multisensory system. Learning and adapting to the habits of this group of people is achieved by introducing artificial neural networks (ANNs) to the output of this system. Thus, any sudden change can be analyzed. The multisensory system along with the ANN methodology used for learning can secure all parts of a complete environment for people with disabilities.

![Typical neural network structure](image)

**Figure 6.8. Typical neural network structure**

| Supervised neural network | – Attempts to predict a specific quantity  
| – Has training examples with labels  
| – Can measure accuracy directly |
| Unsupervised neural network | – Attempts to understand the data  
| – Looks for structure or unusual patterns  
| – Not looking for something specific  
| – Does not require labeled data  
| – Evaluation usually indirect or qualitative |
| Semi-supervised neural network | – Uses unsupervised methods to improve supervised algorithms  
| – Usually few labeled examples and a lot of unlabelled data |

**Table 6.2. Description of different neural network algorithms**

### 6.7.3. Deep neural networks used in human activity recognition

Lately, there has been developing revenue in deep learning techniques. It has become a basic exploration region in human activity recognition, natural language processing, machine interpretation and environmental monitoring (Guo et al. 2014).

Deep learning is an overall term for neural network methods which depend on taking in portrayals from raw data and contain more than one hidden layer.
The network has numerous layers of non-linear data processing for feature extraction and change. Each progressive layer utilizes the output from the previous layer as input.

Deep machine learning algorithms include restricted Boltzmann machines, auto-encoders, sparse coding, convolutional neural networks and recurrent neural networks (Figure 6.9). These deep learning strategies can be stacked into various layers to frame deep learning models that give improved framework execution, adaptability, robustness and eliminate the need to rely upon conventional handcrafted features (Nweke et al. 2018). These methods are applied to activity of daily living (ADL) (Hassan et al. 2018), like locating and detecting posture in Abdel-rahman et al. (2009), recognizing gestures activities of Alzheimer's, and diagnosis of emotional state for elderly people in Ravi et al. (2016). Deep learning methods are used also in automatic detection of activity of daily living (ADL) (Wang et al. 2016; Gu et al. 2018), health rate analysis during intensive sports activities and health monitoring (Jalal et al. 2017; Nweke et al. 2019), and representation of energy-related, health monitoring smart homes (Jianbo et al. 2015). Moreover, some methods go deeper to predict the relationship between exercises and sleep patterns, automatic pain recognition during strenuous sports activities, energy expenditure estimation, and tracking of personal activities (Hammerla et al. 2016). In addition, other applications used deep learning algorithms in HAR like: model temporal patterns in activity of daily living (ADL), progressive detection of activity levels, and falls and heart failure in the elderly (Ordóñez et al. 2016).

Fang and Hu (2014) proposed a deep learning calculation to perceive human activity. They believed the deep belief networks (DBNs) worked using restricted
Boltzmann machines in the research. They additionally contrasted their outcomes and HMM and NBC.

Oniga and Suto (2014) interpreted the signs obtained from speed increase sensors utilizing a few artificial neural networks (ANN) algorithms. Zhang et al. (2015) combined HMM and DNN models to perceive activity. Be that as it may, there is at present no preferred deep learning method for human activity. This is likely because of the changeability of human practices, activities performed, types of sensors utilized and includes selection received.

The calculations of neural networks are based on modeling of the characteristics of the human brain. This leads to a relationship being formulated between the input and output variables on the basis of the observable data. The general model of the neural network consists of:

– examining a process in neurons;
– data interaction;
– multiplying the weights of connections for data transferred from one neuron to another to solve problems;
– calculating the output using the enable function at input.

The different categories of the neural network are shown in Figure 6.10.

*Figure 6.10. Feed-forward and feedback network architectures*
6.8. Anomaly detection in smart homes

6.8.1. What are anomalies?

There are several definitions for the anomaly.

An anomaly refers to any phenomenon that deviates from what is considered normal\(^6\).

Anomalies are data patterns that do not conform to a well defined notion of normal behavior (Chandola et al. 2007).

Anomaly detection or outlier detection is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data (Arthur and Erich 2017).

6.8.2. Types of anomaly

Anomalies are classified into three types.

Point anomalies: this is the case where an individual data instance is considered abnormal compared to the rest of the data.

Contextual anomalies: this is an anomalous data instance in a specific context; it is also called a conditional anomaly (Song et al. 2007).

Collective anomalies: when a collection of linked data instances is abnormal with respect to the dataset, this is the case of a collective anomaly.

6.8.3. Categories of anomaly detection techniques

An instance of data is said to be normal or abnormal according to the labels associated with this instance. Anomaly detection techniques operate in one of three modes, depending on the availability of labels.

Unsupervised anomaly: for an unlabeled dataset, unsupervised anomaly detection techniques assume that the majority of instances in the dataset are normal and look for instances that do not match the rest of the data.

---

Supervised anomaly: detects anomalies in a dataset that has been labeled as “normal” and “abnormal” and involves the training of a classifier.

Semi-supervised anomaly: this detection technique assumes that the training data has instances labeled only for the normal class. For this, these techniques are more widely applicable than the supervised techniques.

6.8.4. Related work of anomaly detection in smart homes

The detection of anomalies has been the subject of several studies in the last ten years. In what follows, we present some works that have dealt with the problems of anomaly detection.

In Stenudd (2010), the authors describe the concept of anomaly detection, used to monitor the behavior of the system to detect normal events from anomalies. It is very useful for security in smart systems, especially for intrusion detection. It is done by comparing the current behavior of the system with previously-stored normal behavior. The anomaly detection helps in detecting insider attacks, is difficult for an attacker to set an alarm off, able to detect known and unknown types of attacks and it fits all deployment environments. But, the authors suggested training the system before deployment to determine the normal behavior, the system generates a false alarm and sometimes users can gradually train the system to accept anomalous behavior as normal.

The research (Jakkula and Cook 2011) aims to detect anomalous events or actions in a smart home. The model starts with environmental sensing by collecting data from inhabitants and their environment to model it. The captured events are associated with timestamps, and then they used with the history of recognized sensor events to reveal patterns with frequent activity. The collected data contains five parameters: date, time, sensor ID, message and annotation. The system uses the one-class support vector machine (OCSVM) which is quite popular for anomaly detection problems. The authors proposed a solution for this problem by estimating a function $f$ which is positive on $S$ and negative on the complement $S$. The algorithm can be summarized as mapping the data into a feature space $H$ using an appropriate kernel function and then trying to separate the mapped vectors from the origin with maximum margin. Once transformed to a different space, the data points which are closer to the origin are identified as an anomaly and reported (Figure 6.11). The research was the initial step in the anomaly detection system and used to start from a new perspective with advanced features.
In Ramapatruni et al. (2019) the authors work on a smart home equipped with many sensors. They presented some anomaly detection models for smart home security. The sensors are connected to the Internet through a wireless router and the data from the home. Data is sent to the gateway. The installation is modulated to add a data set collection machine, by configuring a port on the router dedicated to copying all the packets sent through the gateway to the data collection machine. The idea started by creating a model using hidden Markov models (HMMs) (Rabiner 1989) to learn common behavior in the smart home. The model takes into consideration many parameters like the number of the state as an N and the set of states represented by S corresponding to each sensor individually when used. Then, the different sensors are named with the capitalized first letter of each name like the min door sensor named as (MD). The value of sensors is on or off. Also, the M parameter is the number of unique observations possible for each state in S. The set V denotes the set of all possible observations in it. The model starts the observation under two conditions: the first one is evaluating through the general condition, the second is abnormal detection. The first condition is during the presence of an individual in the house who uses the sensor in real-time. The results show 96.8% of data values were stated as normal behavior. The net experiment performed using modified k-fold cross-validation to determine the efficacy of their approach in a general setting. The accuracy ranged from 95% to 98%. The second condition tested was where there is no user in the smart home. The HMM model detected 97% of the attack anomalies.

Figure 6.11. Data points identified as anomaly detection
In Yamauchi et al. (2020), they work on a new model of anomaly detection related to human behavior by focusing on a specific pattern of usage for users, in this research the work was done on smart home conditions. The anomaly detection learns from the repeated pattern, for example for a specific user on a cold day they turn the heater first then turn the humidifier then the model will detect another behavior if the order of action is changed. The model starts by learning events and then creates the sequence. After this, the model starts monitoring the sequence of events that happen and compares it to the learned one. The model learns through three phases: learning model, learning user behavior and detection. The learning model depends on the condition and user behavior. The first step is to define conditions, the sensor reading and place the condition in the table. In addition, for every condition, they stored the related human behavior. Now learning user behavior is done on the gateway by sorting sequences of events and differentiating between events from the users and events from guests that may be in the same house with the users. This is done by focusing on frequently repeated events that are most likely done by users. In the last phase, “detection”, the gateway recognizes the executed operations and then compares them to the learning events. After the three phases, the model can detect whether the behavior or the events is normal behavior or an attack. The tested result shows a 90% pass in detecting the attack.

In Fu et al. (2021), the research focuses on appified smart homes, a term that refers to the smart home that uses many IoT devices and is connected to a different platform to control events in homes using smart applications. They proposed a Home Automation Watcher HAWatcher, which relies on semantic and logs events and then creates a hypothetical correlation according to semantic information and verifies them with events logs. For example, human activity changes device state then the device state also reflects human activities. The proposed model aims to determine the malfunctions in IoT devices if they are related to the network or technical part and is known as cyber, or physical which is related the device itself. Nevertheless, the model must detect whether it is a malfunction or an attack on the IoT devices. The model executes two steps: extract semantic from the smart app then converts it to correlation. They designed a shadow execution engine for anomaly detection. The role of this engine is to get access to internal device data and perform the tests on incoming events. The model faces some limitations such as deviation in human activity, false alarm rate and an attacker with knowledge of correlations, however it detected 62 real-world anomaly cases with high accuracy.
In Gassais et al. (2020), the authors propose a new framework for intrusion detection that combines machine learning, the space of the user and kernel to detect intrusion in smart devices. The framework installs a whole new infrastructure consisting of sensors, actuators and an analysis system. Smart devices running the tracers and the analysis system detect anomalies by aggregating the collected traces. The framework triggers an alarm when intrusion is detected. It also can be developed to correlate traces and take actions to prevent the intrusion. The framework was tested on many algorithms and the results show different efficiency on different algorithms. It is very useful, however for detecting attacks on the system especially in eavesdropping attacks which are very difficult to detect using network information only.

The Internet of Things allows devices and sensors in a smart environment (such as smart homes) to communicate with each other and share information between platforms. However, the IoT has proven to be susceptible to security vulnerabilities. It was therefore necessary to develop solutions capable of detecting anomalies and solving security problems. Because of this, in this section, we presented some studies that attempted to provide an overview of anomaly detection research.

6.9. Conclusion

In this chapter, we have highlighted the importance of cyber security in home automation using IoT devices, as well as the problems related to the security of the different devices connected in smart homes. The main challenge of smart home systems is the ability to adapt to the user by providing enhanced comfort, control and security.

One of the most commonly-used methods for learning in a smart home system is neural networks. These are enhanced by deep learning which allows the use of a vast amount of data with minimal storage cost and has the ability to learn without being explicitly programmed. In this chapter, we have also explained the relationship between artificial intelligence, machine learning and deep learning and how they differ. We have highlighted the importance of deep learning applications in cyber security and human activity recognition in smart homes using neural networks and deep learning.

In a smart home, many devices are connected to the Internet and therefore can experience problems and can be the target of cyber attacks. These attacks can cause serious problems and harm users. For this reason, we have presented several methods for detecting anomalies and attacks in smart homes.
6.10. References


IOT, Deep Learning and Cybersecurity in Smart Homes: A Survey


sTiki: A Mutual Authentication Protocol for Constrained Sensor Devices

Corinna SCHMITT\textsuperscript{1}, Severin SIFFERT\textsuperscript{2} and Burkhard STILLER\textsuperscript{2}

\textsuperscript{1}Research Institute CODE, Universität der Bundeswehr München, Neubiberg, Germany
\textsuperscript{2}Communication Systems Group (CSG), Department of Informatics IfI, University of Zürich UZH, Switzerland

Today, a large number of use cases exist for the Internet-of-Things (IoT) and Wireless Sensor Networks (WSN), such as home automation, ambient assisted living, eHealth, and logistics (Romeo 2016). For certain use cases it is desirable to make sensitive data (e.g. medical information or personal address) globally accessible (a) to authorized users only and (b) to data processing units through the Internet. Even seemingly inconspicuous data, such as the energy consumption measured by a smart meter, can lead to potential infringements on the users’ privacy, e.g. by allowing an eavesdropper to conclude whether or not a user is currently at home.

From an industry perspective, there is a pressing need for security solutions, especially for the transmission of sensitive data, and access to it has increased. Due to leaks of such information to the public, end-users in the private sector have also demanded security for their data and require privacy support and data ownership rights. From a legal perspective, this is addressed by the EU General Data Protection Regulation (GDPR; Regulation (EU) 2016/679) (European Parliament and Council of the European Union 2016). Regarding the infrastructure of the IoT, including devices with different amounts of resources, security risks are aggravated by the trend toward...
a separation of sensor network infrastructure and applications (ETSI 2010; Leontiadis et al. 2012). Therefore, a true end-to-end security solution is required to reach an adequate level of security for IoT. Protecting data once it leaves a local network is not sufficient, because it may reach the final destination after many jumps and via an uncontrollable network.

However, the IoT is no longer limited to servers, routers, and computers. The IoT also includes constrained (tiny) devices – sensor nodes – limited in memory (app. 10–50 kByte RAM and 100–256 kByte ROM), computational capacity and power (a few AAA batteries) (Bormann et al. 2020). Those limited resources still demand end-to-end security support as requested by data owners, including the mutual authentication of communication partners (source and destination – sensor node, gateway, aggregator), which requires individual key agreements and a secured communication solution to be built between them. When explicitly depending on the resources of these devices, performing authentication and key agreement is challenging, because memory, computational capacity and energy are scarce. Furthermore, a deployed network may be dynamic, such that nodes dynamically join or leave, and require updates in security. Thus, a light-weighted solution is required, as represented by the described solution “sTiki” working under Contiki 3.0.

7.1. Introduction

Due to the growth of the Internet and the diversity of devices now available, the Internet-of-Things (IoT) is gaining a lot of attention. The IoT used to be limited to Peer-to-Peer (P2P) architectures (Gerke et al. 2003), networks, dedicated applications (Mischke and Stiller 2003) and devices, such as servers, computers, and routers. However, the IoT now includes wireless sensor devices that form an individual Wireless Sensor Network (WSN). Those devices present a challenge for developers, because they are limited in memory, energy and computational capacity. In order to connect them to the Internet, they must support IP (Internet Protocol), which is often provided by using an IPv6 implementation called 6LoWPAN (Shelby and Bormann 2011).

The topology of WSNs can range from star topologies to pure P2P topologies, but a combination of both is employed in WSN deployments. This means that the network consists of Full-Function Devices (FFD) and Reduced-Function Devices (RFD). Both of these types of devices can support different functionalities depending on their location within the WSN, ranging from simple data collection and forwarding to pre-processing. The communication between devices in a WSN is performed wirelessly, using the UDP (User Datagram Protocol). The transmitted packet size is limited (e.g. 127 Byte), however existing IPv6 implementations, like 6LoWPAN, support packet fragmentation and compression in order to connect such limited devices to the IoT (Karl and Willig 2007). While many use cases for the IoT involve
the collection and transmission of sensitive data, many deployments currently do not protect this data through suitable security schemes (Sen 2009). Different end-to-end security schemes were built upon existing Internet standards, specifically the Datagram Transport Layer Security (DTLS) protocol, but are not applicable to WSNs due to the use of constrained devices with especially limited memory resources. By relying on an established standard, existing implementations, engineering techniques, and security infrastructures can be reused to enable a security uptake from application developers (Kothmayr et al. 2013).

The challenge is now to bring standards-compliant security to resource constrained sensor nodes in an end-to-end security architecture, fulfilling the request to be light-weight. While the intended solution must satisfy the paradigm of end-to-end security, it has to be based on standards and support mutual authentication (e.g. Luk et al. 2007; Kothmayr et al. 2013; Lowack 2013). It should be taken into account, that the new solution must be able to process the basic functionality of the existing implementation in parallel, especially when gathering data, transmitting it using the TinyIPFIX (Schmitt et al. 2017) format and supporting aggregation in the network (Sgier 2016, 2017). Furthermore, flexibility concerning network updates (e.g. node addition or deletion) is key in order to establish efficient and secure communication within the network.

Such a light-weight solution that has been developed is “sTiki”, working under Contiki 3.0 as an operating system. A key server offers authentication and dynamic node management, which were integrated into the server component on the gateway. Symmetric encryption provides data integrity and a message counter guarantees the freshness of the messages received. For encryption, a multitude of algorithms were considered and compared, with AES-128 (Advanced Encryption Standard) coming out on the top. Finally, confidentiality is achieved by using a Message Authentication Code (MAC). The implementation performed consumes about 4.5 kB of ROM and around 400 B of RAM, which allows sTiki to run on very constrained devices. sTiki was demonstrated to work and was tested successfully in multiple setups, including scenarios where certain nodes could not support sTiki. Combining sTiki with unencrypting nodes does work, even in case of heterogeneous hardware. It is, obviously, not possible to move from encrypted nodes back to unencrypted ones.

The remainder of this chapter1 is structured as follows: section 7.2 illustrates the Internet’s history leading to the current connected world, while section 7.3 presents insights of security for the IoT, and section 7.4 paves the way toward the security protocol “sTiki”, developed for constrained devices. Section 7.5 covers the constraints

---

1. This chapter is based on the Bachelor Thesis by Severin Siffert, University of Zürich UZH (Siffert 2018), who was part of the SecureWSN project (see: https://www.csg.uzh.ch/csg/en/research/SecureWSN.html) lead by Corinna Schmitt during her employment at the University of Zürich UZH and included in her Habilitation Thesis (Schmitt 2019).
of developing software for sensor networks, the architecture chosen, and the choice of AES. The implementation of sTiki is discussed in two environments. Finally, section 7.6 evaluates the implementation and compares it to existing implementations, before conclusions are drawn.

7.2. Definitions and history of IoT

During the late 1970’s the idea emerged to interconnect different networks in order to enhance the communication between universities and research facilities. The purpose of this connection was two-fold: (1) a simple exchange of knowledge using file sharing and (2) using the computational capacity of computer centers worldwide. This approach resulted in the Arpanet. Over the next decades more networks were connected, building the Internet, which was commercialized in 1990. As described in Kurose and Ross (2016) the Internet is defined as a network of networks with a specific infrastructure. Many networks operate as Autonomous Systems (AS), with one or more edge devices (gateways, routers and access points) responsible for forwarding traffic between different ASes Kurose and Ross 2016). Compared to other network participants, such as computers or servers, only a reduced OSI (Open Systems Interconnection) protocol stack is supported. This stack enables the forwarding of messages within an AS and between different ASes, even translation between different communication standards used on the physical layer (PHY) are possible.

During the last few decades the connectivity of devices used increased, due to stakeholders’ requests to be connected all over the world using manifold devices, and with them manifold communication technologies. This is referred to as the Internet-of-Things (IoT). The term IoT was first coined by Kevin Ashton at a presentation at Proctor & Gamble in 1999 (Ashton 2009). According to Ashton, the IoT is a network of connected physical and fingerprintable objects (things) with a virtual representation in an Internet-similar structure. This definition still holds today, with the only exception that the connected device characteristics have changed over time. In the beginning only routers, hat ways, servers, and PCs were intended to be part of IoT, but new devices with specific and different characteristics have since been added. These new devices show the following characteristics (Schmitt 2019): (a) device size ranging from coin size to commonly used notebooks over smart devices, smart phones, and tablets, (b) devices support communication standards such as 3G to 5G and wireless communication, (c) devices are resource-constrained concerning energy, memory, and computational capacity, and (d) devices potentially require application specific (e.g. sensor devices, navigation systems, or alarm system) implementations and deployments. The existing diversity of devices shows that establishing connectivity between them may be challenging due to their characteristics, and may become even more complex when data formats, data transmission frequency, and requested support by stakeholders in terms of security and privacy are needed.
Similar to the aforementioned diversity in device characteristics common for IoT devices, is the diversity in terms of the definition of “IoT” itself. Research and industry refer to the aforementioned situation and the resulting network under the term IoT, as indicated in Figure 7.1. Deeper investigations show that no formal definition for the IoT exists overall, which makes discussions and comparisons of approaches and developments challenging. The following definitions exist:

International Telecommunication Union (ITU) Standard Y.4000 (ITU 2016):

Internet of Things (IoT): A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. NOTE 1 – Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled. NOTE 2 – From a broader perspective, the IoT can be perceived as a vision with technological and societal implications.

EU-Parliament Briefing (Davis 2016):

The Internet of Things (IoT) has been defined in a number of different ways. Generally speaking, it refers to a global, distributed network (or networks) of physical objects that are capable of sensing or acting on their environment, and able to communicate with each other, other machines or computers. Such “smart” objects come in a wide range of sizes and capacities, including simple objects with embedded sensors,
household appliances, industrial robots, cars, trains, and wearable objects, such as watches, bracelets or shirts. Their value lies in the vast quantities of data they can capture and their capacity for communication, supporting real-time control or data analysis that reveals new insights and prompts new actions. As in the case of many emerging technologies, different experts may use different terms to refer to similar or overlapping concepts. Machine to machine (M2M) processing emphasizes the sharing of data and processing that takes place between these devices. On the other hand, the Internet of Everything explicitly includes people as participants in this global network. Ubiquitous computing emphasizes the fact that network and computing resources are available almost everywhere, whereas pervasive computing highlights the fact that processors are embedded in everyday objects all around us.

Internet Engineering Task Force (IETF) (Lee et al. 2010):

The basic idea is that the IoT will connect objects around us (electronic, electrical, non electrical) to provide seamless communication and contextual services provided by them. The development of RFID tags, sensors, actuators and mobile phones makes it possible to materialize things that interact and co-operate with each other to make the service better and accessible anytime, from anywhere.

Apart from these definitions, each developer, industry and service provider uses its own definition of the IoT depending on their specific settings, requirements, and views (International Data Corporation 2014; SAP 2014; 451 Research 2015). In general, those definitions focus on the stakeholders involved by grouping them by purpose (e.g. offered services, standardization organizations, service or platform providers, network infrastructure provides, protocol and device developers) (International Data Corporation 2014).

The cluster of European Research Projects on the Internet of Things (CERP-IoT) (Sundmaeker et al. 2010) states:

The Internet of Things (IoT) is an integrated part of the Future Internet and could be defined as a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols, where physical and virtual “things” have identities, physical attributes and virtual personalities, use intelligent interfaces, and are seamlessly integrated into the information network. In the IoT, the “things” are expected to become active participants in business, information and social processes, where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information “sensed” about the environment,
while reacting autonomously to “real/physical world” events and influencing them by running processes that trigger actions and create services with or without direct human intervention. Interfaces in the form of services facilitate interactions with these “smart things” over the Internet, and query and change their state and any information associated with them, taking into account security and privacy issues.

7.3. IoT-related security concerns

Due to the connectivity of devices in the IoT, the data amount continuously increases. A report by CISCO (Bhaiji 2008) stated that IP networks are growing exponentially and due to today’s device manifoldness and application variety, networks are becoming more and more complex, with new challenges to run and manage them arising at the same time. This assessment has been confirmed by continuous statistics from various providers (e.g. statistica and CISCO (CISCO 2019)). Thus, the classic network infrastructure undergoes an evolution, introducing security concerns, which continuously result in changes in security paradigms (i.e. CIA triad – confidentiality, integrity, and availability (Andress and Winterfeld 2014; Summers and Tickner n.d.)) due to applications, environments and end-user requests linked with the resources of included devices in the network and the communication standards used.

These changes in security paradigms become obvious when extending common IP networks with constrained networks building the IoT today. Constrained networks show common characteristics with IP networks, especially in relation to security fundamentals (sometimes also named cryptographic properties), following the CIA triad – confidentiality, integrity, and availability – and enhancing it with data freshness and authentication (Boyd and Mathuria 2010). Whereas, confidentiality ensures that data is only available to those that are authorized to obtain it, integrity ensures that no message can be altered by an entity as it traverses from the sender to the recipient, and availability ensures that the service of a constrained network is always available, even in the presence of internal or external attacks (e.g. Denial-of-Service (DoS) attack). This triad is enhanced by data freshness, which implies that the data is recent and ensures that no adversary can replay old messages and authentication, which ensures the identity of the communication partners. Literature adds three additional security fundamentals (Boyd and Mathuria 2010; Eckert 2014): data origin authentication guarantees the origin of data, in order to archive entity authentication in protocols and establish keying material. Non-repudiation ensures that entities cannot deny sending data that they have committed to. Finally, resilience represents the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation.

In addition, further security fundamentals become essential for the assumed use case of constrained networks, due to their deployment options (random, fixed,
partly fixed) (Karl and Willig 2007). *Self-organization* is required in a constrained network due to the dynamic nature of it, which makes it impossible to deploy any pre-installed shared key mechanism between several nodes. *Secure localization* is also necessary to locate each constrained device accurately and automatically if faults should be detected. Finally, *time synchronization* is needed for collaborative constrained networks or certain security mechanisms, such as periodic updates of keying material.

![Figure 7.2. Attack types in IP networks (Schmitt 2019)](image)

As constrained networks like Wireless Sensor Networks (WSN) are part of the Internet and are constructed out of constrained devices that usually have limited resources, they represent a criticality in the IoT. As highlighted in Figure 7.2, common attacks on constrained devices with a sensor device are (a) attacks against service integrity, (b) attacks on secrecy and authentication, and (c) attacks on authorization and authentication (i.e. Denial-of-Service (DoS) attacks) (Shi and Perrig 2004; Wang *et al.* 2006; Boyd and Mathuria 2010; Eckert 2014). Normal defense mechanisms against attacks can hardly work or not work at all for them due to the devices’ constraints. Before addressing defense and prevention mechanisms, potential
vulnerabilities must be determined. Hence, the following provides an overview of the most relevant security analysis guidelines, followed by the description of a threat modelling for the security analysis of solutions available. Based on this knowledge gained about the IoT, the security insights combined with security expectations lead to the definition of sTiki.

### 7.3.1. Security analysis guidelines

Before deploying protocols, algorithms, workflows, frameworks, and services or integrating them into commercial services and systems, they need to undergo a security analysis to check if they follow the security paradigms applied and requested by end users. For a security analysis, different guidelines apply to different application areas and developments (see section 7.3.1). For IoT networks and constrained networks especially, such analysis is essential to determine the specific service with respect to sensitive data from IoT devices.

The National Institute of Standards and Technology (NIST) published several recommendations, the most important of which are the “Information Security Handbook: A Guide for Managers” published 2006 and the “Technical Guide to Information Security Testing and Assessment” from 2008. The first recommendation provides:

> a broad overview of information security program elements to assist managers in understanding how to establish and implement an information security program. The topics within this document were selected based on the laws and regulations relevant to information security […] The purpose of this publication is to inform members of the information security management team (agency heads; chief information officers (CIOs); senior agency information security officers (SAISOs), also commonly referred to as Chief Information Security Officers (CISOs); and security managers) about various aspects of information security that they will be expected to implement and oversee in their respective organizations. In addition, the handbook provides guidance for facilitating a more consistent approach to information security programs across the federal government. Even though the terminology in this document is geared toward the federal sector, the handbook can also be used to provide guidance on a variety of other governmental, organizational, or institutional security requirements (Bowen et al. 2006).

In comparison, the second recommendation offers:

> a guide to the basic technical aspects of conducting information security assessments. It presents technical testing and examination methods and
techniques that an organization might use as part of an assessment, and offers insights to assessors on their execution and the potential impact they may have on systems and networks. For an assessment to be successful and have a positive impact on the security posture of a system (and ultimately the entire organization), elements beyond the execution of testing and examination must support the technical process. [...] The information presented in this publication is intended to be used for a variety of assessment purposes. For example, some assessments focus on verifying that a particular security control (or controls) meets requirements, while others are intended to identify, validate, and assess a system’s exploitable security weaknesses. Assessments are also performed to increase an organization’s ability to maintain a proactive computer network defense. Assessments are not meant to take the place of implementing security controls and maintaining system security.

Following these recommendations, the target organizations of developments are requested (1) to establish an information security assessment policy, (2) implement a repeatable and documented assessment methodology, (3) determine the objectives of each security assessment, and (3) tailor the approach accordingly, analyze findings and develop risk mitigation techniques to address weaknesses (Scarfone et al. 2008).

Besides NIST, further investigations are ongoing to strengthen security analysis and establish standards. One of the most well known ones is the so-called “OWASP Application Security Assessment Standards Project”. Its mission is:

to establish common, consistent methods for application security assessments standards that organizations can use as guidance on what tasks should be completed, how the tasks should be completed and what level of assessment is appropriate based on business requirement. [...] The final goal is to integrate a set of OWASP projects into an Application Security Assessment process in order to define a model which can be used by an organization to provide application security through OWASP standards. (OWASP Application Security Assessment Standards Project 2014)

Due to the continuous growth of technology and attack possibilities (see Web Application Security Consortium (2004) and The Open Web Application Security Project (2019)) the specification of security analysis is undergoing updates periodically, in order to recommend effective prevention and countermeasures. Currently, the following information security assessment types are commonly used and recommended to define and design appropriate solutions for applications:

– A vulnerability assessment counts towards the technical assessments. Its purpose is to yield as many vulnerabilities as possible in an environment, along with severity and remediation priority information.
– **Penetration tests** count towards the technical assessments as well and target a specific goal to identify vulnerabilities. Common target vectors are stealing customer data or gaining administrative access to an infrastructure.

– An **audit** can either be technical and/or document-based. It focuses on how an existing setup or configuration matches standards that need to be followed. Classic examples are who has access to a server, how often access rights are checked and if required revoked, and which backup procedure is implemented.

– A **risk assessment** is recommended to be performed periodically and aims to determine what the current level of acceptable risks is, measuring the current risk level, and then determining what can be done to bring these two in line where there are mismatches.

– A **threat assessment** focuses more on physical attacks than on the technology used. The main purpose is to determine whether a threat is credible or not and to identify how many resources are required to address the threat.

### 7.3.2. Security analysis by threat models

In addition to the aforementioned guidelines and recommendations mentioned it also became practice to discuss the security of protocols in the context of a threat model. The result of a threat model depends on the exact specification of the attacker analysis, and it is essential to be precise. The most used approach is the Dolev–Yao security model (Dolev and Yao 1983), which gives a formal definition of the strongest possible attacker. It is often used in the security evaluation of protocols and assumes that the attacker is in control of all of the communication channels. In particular, the attacker may carry out any of the following (Dolev and Yao 1983):

– **Eavesdropping** assumes that the attacker may eavesdrop on any communication in the network. Further it can be assumed that they have recorded all prior communication between any two nodes.

– **Message insertion** assumes that the attacker knows the protocol specifications and can create and insert new messages at will.

– **Message delay and delete** assumes that the attacker may delay any message in the network, or even delete it.

– **Message modification** assumes that attackers can modify intercepted messages.

– **Replays and out-of-order messages** assumes that the attacker can replay any previously sent messages, and they can also forward them out of order.

– **Attempt at impersonation** assumes that the attacker may attempt to act under the identity of another principal.

– **Old session keys** assume that the attacker is in possession of session keys from older protocol sessions.
Disabling the network assumes that the attacker may flood the network with messages, or partially or completely disable it ("cut the wire") at any time. This is a capability against which a protocol cannot take measures. Protocols can only be designed to raise the barriers for the attacker as high as possible.

Note that the attacker may be a legitimate user of the network (i.e. they may act under their own identity). The attacker is, however, bounded by the strength of the cryptographic primitives’ encryption and signatures. They cannot decrypt messages and they cannot forge signatures or message authentication codes without the knowledge of the corresponding key. This effectively results in secure channels on which the attacker cannot modify messages without tampering being detected, but they may still be able to suppress messages and interact with the normal protocol flow.

The rationale of using this model is as follows: first, it is one of the standard models in the security community, and is well understood. Second, it is reasonable to design a protocol that is secure against the strongest possible attacker because in many cases we cannot be entirely sure of the capabilities of an attacker. Third, current model checkers for cryptographic protocols, such as AVISPA or Scyther, use the Dolev–Yao model internally (Cremers et al. 2009). Model checkers have become an important method for evaluating the security of protocols and finding attack vectors. It is important to observe that the Dolev–Yao model is a model for the formal analysis of a given system. It restricts the attacker to attack vectors within the system. Therefore, it does not cover attacks outside of the system, like social engineering or errors in an implementation that lead to secret information being compromised.

The Dolev–Yao model was implemented into tools to automate the analysis of security protocols. The most common used tool nowadays is AVISPA\(^2\) (Automated Validation of Internet Security Protocols and Applications). Its design is inspired by common software design following the push-button strategy for the automated validation of Internet security sensitive protocols and applications. It gives the developers of protocols the possibility to specify their protocols and assumed security properties with the help of a modular and expressive formal language. The tool integrates different backends, allowing the implementation of a variety of state-of-the-art automatic analysis techniques, which are applied on the developer’s input.

### 7.3.3. sTiki’s security expectations

As depicted above, four security fundamentals exist, but not all may be needed or desired: (a) confidentiality, (b) integrity, (c) authentication, and (d) freshness. Authentication and integrity can both be achieved by using a cryptographic

---

2. See: http://www.avispa-project.org/.
checksum, commonly called a Message Authentication Code (MAC). Freshness is usually provided by timestamps, nonces or counters. Confidentiality is achieved by encryption. In order to achieve all four, decisions must be made: the first decision is made between symmetric and asymmetric encryption, as there are arguments in favour of both. As in many decisions with WSNs, this one too is between efficiency and security. Asymmetric ciphers win in security, but take significantly more resources (Mohd et al. 2015). For example, ECC (asymmetric) is about 100–1000x slower than AES (symmetric) (Potlapally et al. 2003; Eisenbarth et al. 2007). For the planned use cases, symmetric will be fine and will also allow a better comparison to Lowack (2013). The second decision is made between block and stream ciphers. Block ciphers encrypt blocks of a fixed size, whereas stream ciphers encrypt data of arbitrary length. Research about lightweight cryptography focuses mainly on block ciphers because they can also be used for computing MACs, and if necessary, can be turned into stream ciphers by using CBC (Chain-Block Chaining) or Counter mode (Mohd et al. 2015). As a result of this, the most high quality lightweight ciphers are block ciphers and when using one, the implementation of an additional cipher for computing MACs can be omitted, which is also why a block cipher is used here. Mohd et al. (2015) compared many lightweight symmetric block ciphers for various criteria. In addition to these ciphers compared, a new ultra lightweight block cipher called QTL (Li et al. 2016) was considered, but Çoban et al. (2017) and Sadeghi et al. (2017) have proven it to be insecure. Lightweight ciphers are a specialized category of encryption algorithms that try to find a good trade-off between resource consumption and security. They are commonly used in WSNs, Wireless Body Area Networks (WBAN), and other medical devices (Mohd et al. 2015). The power consumption is especially a concern. Because of the various security requirements of different applications (controlling a pacemaker vs. a home temperature monitoring system), different algorithms and variations exist.

Comparing different ciphers reveals that no universally accepted metric exists for measurements; security is not a well-defined term (Mohd et al. 2015). In addition, new and innovative ways of attacking highly rated algorithms may be discovered and render the ratings useless. Even a metric that does not consider security, like efficiency (e.g. defined as energy required per encrypted byte), fails at being a fair comparison, because it can be gamed in various ways (Badel et al. 2010). Instead, AES shows up close to the top for almost all metrics used in Mohd et al. (2015) and Potlapally et al. (2003), which makes it a relatively obvious choice. Its major problem is its relatively large memory footprint. But many hardware boards (including OpenMote) offer a hardware-implementation of AES, which requires almost no additional memory. The most popular implementation is AES-128. Based on the results as of Mohd et al. (2015), Tea/xTea would have been the second choice.

Even though the parameters of the hardware implementation of the OpenMote B platform selected (see section 7.4.1) cannot be changed, it is still required to understand the impact on energy consumption, in order to judge potential alternatives. The main parameters of block ciphers are key-size, block-size, and the number of
rounds performed. The increase in energy consumption is roughly linear to key-size and the number of rounds (Potlapally et al. 2003). The main parameters, however, do not have the largest impact on energy consumption. Instead, the mode of operation has an impact that is two to three times larger (Potlapally et al. 2003). This is because the different modes run different procedures to determine the key for the next encryption step, which can be a substantial effort.

7.4. Background knowledge for sTiki

This section covers the most important technologies influencing the development of the proposed security protocol sTiki, which is designed for a subfield of IoT, namely WSNs. Such networks consist of just a few or up to thousands of small, very limited computers, normally called nodes or constrained devices. The nodes are usually battery powered and their memory is very limited, often measured in kB. Their purpose is to gather data, for example, about the temperature, humidity or movement of certain objects. Due to the limited memory and and the high probability of failure, the nodes usually send collected data to a device (or device combination) called a sink or gateway, which is more reliable and has more power and memory available (depicted in Figure 7.3). When a node is too far from the gateway, another node can forward the message to another node or to the sink. In certain use cases it is even feasible that a node processes the data (e.g. filtering out measurements that are not needed at the moment or computing an average value), in order to send less bytes or more useful data. The aggregators in Figure 7.3 do exactly that, whereas the collectors only produce measurements. Because the nodes are constrained in resources, they require specialized tools, such as the operating system or protocols shown in the following sections (Akyildiz et al. 2002; Yick et al. 2008; Siffert 2018).

As any protocol design is highly influenced by the hardware and operating system used, a brief overview for sTiki is presented in section 7.4.1. Some of the resources, especially memory, may already be consumed by existing application that need to be further supported by the new “add-on” protocol. This is also the case for sTiki, where a special TinyIPFIX application for data collection is required to be further supported, representing the individual payload that needs to be secured by sTiki. This is also briefly described in section 7.4.1. The identified security expectations for sTiki listed above influenced the selection of existing and light-weighted security protocols.

7.4.1. Application dependencies for sTiki

Here, OpenMotes were used as constrained devices to build the network. They are very small and are a combination of the OpenMote-CC2538 Rev.E and OpenUSB Rev.B parts. The OpenMote-CC2538 mainly consists of the CC2538 processor from Texas Instruments, which has an ARM Cortex-M3 microcontroller with 512 kB
storage and 32 kB RAM. It also has hardware implementations for AES-128 and AES-256. The OpenUSB has a USB port, space for two AA batteries and sensors for light, temperature, humidity, and acceleration on three axes (Tex 2012; Vilajosana et al. 2015; Sgier 2017; Siffert 2018).

![Assumed WSN Setup for the sTiki Implementation (Siffert 2018)](image)

Specialized operating systems (e.g. Contiki (Dunkels et al. 2004), TinyOS (Levis et al. 2005), LiteOS (Cao et al. 2008) or MantisOS (Bhatti et al. 2005)) exist for WSNs, avoiding quick exhausting of the constrained devices in manifold angels, such as Random Access Memory (RAM), Read-Only Memory (ROM), and battery power. Therefore, it is not possible to run Windows or Linux on them. As Contiki is used in this setup, it is briefly characterized here. Contiki is an open source minimal OS written in C, that was created by Adam Dunkels in 2004. It has a modular architecture which is built on top of an event-driven kernel (Dunkels et al. 2004). Whenever an event is triggered, it runs to completion, but can be preempted if necessary. To keep the size as small as possible, threads are only implemented as a library that is included when needed (Farooq and Kunz 2011). In the spirit of saving resources, Contiki offers no way to synchronize the internal clocks, which makes it impossible to implement a multitude of protocols. Contiki can dynamically load and unload code, which allows the running code to be changed remotely and without recompiling (Dunkels et al. 2004). As Contiki is extremely resource efficient, there are only two ways to start code execution: either code can run once the node has powered up, or in reaction to an event happening like a timer running out or a packet arriving. In general, this event-driven nature of Contiki has a big impact on the coding of applications. But here, the impact is limited due to the fact that an encryption protocol mostly works in response to the requests – encrypt or decrypt a message – of the application it supports.
sTiki will secure the application’s payload during transmission in the WSN. To understand the final design of sTiki it is essential to have basic knowledge of the important payload and its format. Thus, a short description is presented here. The payload follows the TinyIPFIX format representing an adaptation of the IP monitoring protocol IPFIX (Schmitt et al. 2016, 2017). The concept of IPFIX became interesting for WSNs as it also applied a push concept of messages in pre-defined intervals. Another advantage is the splitting of measurement and meta data into two small messages – data records and template records – using IDs for cross-referencing and limiting the transmission number of template records, as they stay unchanged during operation. This design leads to a reduction of traffic in the network. Comparing TinyIPFIX to IPFIX further, it can be observed that TinyIPDIX omits a lot of information that is not used in most WSN contexts, thereby reducing the size of these packets.

The implementation of the utilized TinyIPFIX also supports in-network data or message aggregation. Being able to aggregate the measurements is crucial to sending less data. This is important because sending data is one of the most power-intensive tasks a node can perform (Lowack 2013). Depending on the desired form of aggregation, it is possible to save a substantial amount of messages and energy by, for example, sending the average temperature of an entire room over two minutes instead of sending five nodes’ measurements every ten seconds to a far away sink. If every measurement has to arrive at the sink, it is at least possible to perform message aggregation. Message aggregation works by combining multiple payloads into one, which works because TinyIPFIX has a constant overhead per message, regardless of payload size. Before implementing TinyIPFIX into an application it should be considered whether or not it actually makes sense to use TinyIPFIX. When it takes more power and time to aggregate the measurements than to simply forward them to the sink, then TinyIPFIX should only be deployed if network congestion is a bigger problem than the additional power requirements. Both forms of aggregation require that messages can be read by aggregators. If they cannot (i.e. due to encryption), aggregators are no longer able to take advantage of the aggregation. Being able to decrypt traffic on aggregator nodes is a key requirement for the encryption protocol discussed below.

7.4.2. Inspiring resource-efficient security protocols

The three protocols TinySAM (Lowack 2013), MiniSec (Luk et al. 2007), and TinyDTLS (Kothmayr et al. 2013) inspired the sTiki protocol. sTiki relies heavily on TinySAM and uses similar methods and mechanisms and, thus, it is described in more detail compared to the other two protocols.
TinySAM (Lowack 2013) is an application layer encryption protocol that uses any symmetric encryption (here and in Lowack (2013), AES-128 is used) and a key server. There are multiple reasons for those choices. It is implemented as an application, running below other applications, because this makes it as platform independent as possible. The protocol would support asymmetric cryptography with only minor adjustments, but asymmetric cryptography uses significantly more RAM and ROM than symmetric cryptography (Hummen et al. 2014), even though it would offer more security (Mohd et al. 2015). Using a key server is a tradeoff solution. Having a single key for the entire WSN is very susceptible to node capture and can compromise the network in its entirety. Because of that, every link should be encrypted with its own key. But storing the keys for interacting with every other node requires a lot of ROM, renders key distribution extremely complicated, and makes adding new nodes to the network very expensive (Khan et al. 2012). Using a key server is not without problems either, because it forms a single point of failure and is thus a weak point (Bechkit et al. 2012), but in the planned use cases with a single sink node, the same problems exist, even if the key server was not used.

Each TinySAM packet starts with a common header. The header may begin with a magic number, which serves to discriminate between packets that are encrypted with TinySAM from ones that are not. This is necessary in networks that have nodes that have no support for TinySAM. Also in the header is a number specifying the (TinySAM-internal) protocol (e.g. handshake, data transport, or alert) and sub-protocol (i.e. what type of alert) a packet should be forwarded to.

![Figure 7.4. Messages sent in the ANOR handshake (Lowack 2013)](image)

To establish the session key, the ANOR-protocol (see Figure 7.4) is used. ANOR (AN Otway-Rees) (Abadi and Needham 1994) is an improvement over the Otway-Rees key establishment protocol (Otway and Rees 1987). Each node has a key it uses to communicate with and authenticate itself with the key server. To begin the handshake, Node A sends the first message $M_1$ (containing $ID_A$ and a nonce $N_A$) to B. B forwards this message, $ID_B$ and nonce $N_B$ to the key server in the second message, $M_2$. If both nodes belong to the network and may communicate with each
other, the key server generates a session key $K_{AB}$. Then, the session key is encrypted once with A’s initial key $K_{AS}$ and once with B’s initial key $K_{BS}$. Both versions are sent to B in $M_3$, which decrypts the session key with its own initial key. To conclude the handshake, B sends $M_4$ with the encrypted session key to A. By using B as an intermediate station to send the key to A, A only receives the session key after B has received the session key. This proves to A that the handshake was successfully completed and signals that buffered messages can be sent securely, using the just established session key. When working with static keys, sending the same data twice results in the same (encrypted) message being sent twice. This allows attackers to draw conclusions from repetition patterns. To protect against such an attack, a counter mode is used for encryption with a changing initialization vector (IV), resulting in a new key for every message. Because sending data draws a lot of power, the whole initialization vector is not transmitted with every message. Instead, certain bytes of it act as a message counter and only this counter is sent along with the payload. This counter can also be used to prevent replay attacks, where an attacker captures a message and later sends it again. This will be detected because the counter is not increasing from one message to the next. To ensure the data (for example the message counter) has not been tampered with, a MAC is added to almost all of the messages. To calculate the MAC, the session key has to be known and changing only one bit will result in a completely different MAC, making tampered or incorrectly transmitted data easy to detect.

The way TinySAM uses encryption results in all four properties being covered: using MACs produces authentication and data integrity, the counters offer freshness, and encrypting the data during transmission achieves confidentiality. TinySAM does not prescribe a specific cipher. With some minor modifications, even asymmetric ciphers would be feasible. In TinySAM, there are two common exceptional states. In the first case, a node has lost the initialization vector or has detected a packet with the wrong message counter. In that case, it instructs the other node to send a new initialization vector along with the next message. The other problem occurs when the session key was lost on a node, most likely because it lost power for a short amount of time, or because it crashed and had to reboot. If the node sending the message has lost the key, it will initiate a new handshake. The other node will simply overwrite the old session data. If the receiving node has lost the session key, it will relay the command to have the other node initiate a new handshake via the key server. This message is sent via the key server because the key server is the only node that has a key to communicate securely with the other node. If the instruction does not need to be secured, then a malicious node could permanently send ‘missing session’ alerts in the name of any node in the network, shutting down any communication.

**MiniSec** (Luk et al. 2007) is a security solution for WSNs with a version for unicasting and a version for broadcasting messages. Its main goal is low energy consumption, but with as little compromise in security as possible. In MiniSec, each node pair shares two encryption keys, one for each direction. This requires the network
layout to be known in advance and does not allow new nodes to be added easily. The encryption is done with *Skipjack* in *Offset CodeBook* mode, which has the advantage of only requiring one pass over messages to produce the cipher text, as well as the integrity protection, thereby saving a lot of expensive encryption calculations. Message counters are included in the encryption keys to produce differing cipher texts, even when sending the same content repeatedly and to protect against replay attacks (Luk *et al.* 2007). MiniSec achieves all four security fundamentals listed in section 7.3.3. Confidentiality is provided by using encryption, authentication and data integrity are the result of integrity protection and freshness is achieved by using counters that alter encryption keys and initialization vectors. The main disadvantages of MiniSec are the pre-shared keys and its integration in the networking stack of the operating system. The pre-shared keys require a lot of planning before deployment and there is no easy way to switch out a single node in a network. The deep integration into the operating system makes it very convenient to use once it is set up, but the implementation is difficult and very hard to reuse in case it should be ported to a different operating system.

**TinyDTLS** (Kothmayr *et al.* 2013) is an implementation of the Datagram Transport Layer Security (DTLS) protocol on TinyOS for OPAL nodes with a Trusted Platform Module (TPM). The DTLS protocol is an adaption of SSL/TLS, altered to support unreliable communications, such as by UDP. In DTLS, the communication partners can authenticate each other, but do not have to. To do so, they present their X.509 certificates to each other, which will be verified by the certificate authority. Authenticity, therefore, is voluntary, but can be forced by the application. Confidentiality is provided by encrypting the payload and integrity is guaranteed through the use of MACs (Kothmayr *et al.* 2013; Siffert 2018).

The fact that DTLS is a standard protocol is a key advantage for interoperability with other systems. But this compatibility has a steep price: the overhead (compared to TinySAM (Lowack 2013) and MiniSec (Luk *et al.* 2007)) is large and messages are even padded, since DTLS only works with block ciphers. The code size and/or hardware support are also expensive: the handshake uses RSA encryption, the payload encryption AES-128 and the MAC computation SHA1, and to securely store the certificate, a TPM is required. Implementing and performing three different ciphers is expensive for sensor nodes, which only makes DTLS useful for nodes with spare resources. Finally, introducing a certificate authority adds a single point of failure to the system, which can be a high risk when working in harsh environments or with a severely limited power supply. The cost of generating, storing and distributing the certificates should also not be disregarded. The implementation of TinyDTLS uses about 20 kB of RAM and 67 kB of ROM, which is about ten times the amount TinySAM uses (RAM and ROM), or four times the amount of ROM and 25 times the amount of RAM used by MiniSec (Luk *et al.* 2007; Kothmayr *et al.* 2013; Lowack 2013).
7.5. The sTiki protocol

sTiki is designed in a similar way to TinySAM, which is an application layer encryption protocol using AES-128 for the symmetric encryption. As using a single key for the entire network deployment is very susceptible to node capture and can compromise the network in its entirety, a key server solution is combined with sTiki, and as a result, every link is encrypted with its own key. A drawback would be that storing keys for interactions with every node in the network would be memory consuming and make key distribution extremely complicated, especially when nodes are added to the network (Khan et al. 2012). Another drawback is that the key server solution would create a single point of failure in the network that is generally attractive for attackers (Bechkit et al. 2012). But this last drawback already exists, as in the existing network a single-point of failure is in place. This point is represented by the last node in the network, called a sink, which communicates with the server and together with it, represents the gateway (see Figure 7.3).

A sTiki packet always starts with a common header, including the following (Siffert 2018): (a) a magic number, which allows encrypted and unencrypted packets to be distinguished, as some devices (i.e. TelosB) may not have sufficient resources to encrypt data due to very limited resources and (b) a number specifying the sTiki-internal protocol (e.g. handshake, data transport, or alert) and sub-protocol (i.e. type of alert) used, a packet should be forwarded to.

![Diagram](image)

**Figure 7.5.** sTiki’s architecture (Siffert 2018). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip

In order to establish session keys, the ANOR (AN Otway-Rees) (Abadi and Needham 1994) protocol is used, where each node has a key it uses to communicate and authenticate itself with the key server (see Figure 7.4). It is globally assumed that each participating node shares a unique key with the key server S in advance. From this key, two keys are derived during node startup (by encrypting separate hard-coded values), which are then used to communicate with the key server S. There is one key

for computing the MAC, and another for message encryption. This shared key is called the initial master key (IMK) and, thus, S knows all IMKs of all nodes of the network, especially A’s IMK, called K_AS and B’s IMK, called K_BS. When performing the handshake to create the session key K_AB, the following steps are performed and four messages are sent between A, B, and S, as illustrated in Figure 7.4 (Siffert 2018; Schmitt 2019):

1) The handshake is initiated by node A, sending message M_1 to node B. This message includes A’s identification ID_A, the destinations’s identification ID_B, and is encrypted with K_AS a nonce N_A.

2) When node B receives M_1, it first verifies if ID_B in M_1 matches its own ID and ID_A matches the ID of the source node A.
   - If this check fails, the handshake is aborted.
   - Otherwise, B appends its own generated nonce N_B encrypted with K_BS to the message and sends the resulting message M_2 to the key server S.

3) S receives M_2 and checks if the IMKs of A and B are known.
   - If not it aborts the handshake.
   - Otherwise, S generates a random session key K_AB and two tokens T. Every T includes ID_A, ID_B, and the nonce of the appropriated node – either N_A or N_B – and K_AB. The resulting tokens T_A and T_B are generated as soon as S encrypts T with the IMK with A and B respectively.

4) S sends the message M_3 encrypted to B, including T_A and T_B.

5) B verifies M_3 (see Lowack (2013)) and if successfully verified it decrypts its token T_B, stores K_AB, and forwards T_A to A as message M_4.

6) A verifies M_4 (see Lowack (2013)) and if successfully verified it decrypts its token T_A and stores K_AB, completing the handshake.

By using B as an intermediate station to send the key to A, A only receives the session key after B has received the session key. This proves to A that the handshake was successfully completed and signals that buffered messages (e.g. template and data) can be sent securely, using the just established session key. As mentioned in Siffert (2018) using static keys to send the same data twice would result in the same encrypted message sent twice, allowing an attacker to draw a conclusion from repetition patterns. In order to overcome this problem, a counter mode is used for encryption with a changing initialization vector, resulting in a new key for each message (Lowack 2013). As sending large IVs would require a lot of space from the limited Message Transfer Unit (MTU) size, it was decided that only certain bytes of it acting as a Message Counter (MC) were sent along with the payload. This counter can also be used to prevent replay attacks, where an attacker captures a message and later sends it again. The reason for detection is that the counter is not increasing from one message to the next. To ensure that data has not been tampered with, a MAC (see Siffert (2018); Lowack (2013)) that was not mentioned in the handshake steps explicitly, is added to the messages.
7.5.1. Design decisions taken

The sTiki protocol implemented in Contiki version 3 shows four assumptions (Siffert 2018): first, the magic number is included in the header. Second, AES-128 was chosen for encryption. Third, only one ongoing handshake per node is allowed at a time, but several keying materials can be stored, meaning that a node can have several links to neighboring nodes at the same time. Since each handshake takes up to 50 B plus the overhead of searching through them during handling, this consumes too many resources on constrained devices at a time. And finally, a failed handshake step will not be retried, meaning the handshake itself has to be initiated again. With this, blocking the activity of a node is avoided due to trying to resend a message.

As the Contiki operating system is structured in a modular fashion, sTiki was also designed following this paradigm. Modules allow for a simple adaptation, if required, by switching out parts, such as the encryption algorithm, or making smaller changes for update purposes or extensions. A modular structure allows code to be kept local, simplifying maintenance and testing. Figure 7.5 illustrates the modular architecture of sTiki, where the sTiki module is the starting point. As soon as messages are received, the sTiki module checks whether the message is in sTiki format. If not, it is responsible for redirecting messages to the appropriate protocol(s). Thus, the sTiki module functions as a dispatcher delegating the handling to respective modules. Thus, it checks for handshake messages or data messages, delegating them as required. Besides the main module – sTiki module – and protocol handlers (Datatransport module, Handshake module, and Alert module), two helper modules – Sessionstore module and Encryption module – are included in the architecture. These modules perform critical functions or carry state information used throughout sTiki and are integrated in most files in the implementation. The final implementation of sTiki consists of a node (see section 7.5.2.1) and a key server part (see section 7.5.2.2).

Due to memory and power limitations, sTiki needs to be efficient here (Siffert 2018; Schmitt 2019):

The limited RAM and ROM makes it necessary to keep in mind how much of these two resources is still available on the devices, as many decisions favor RAM, ROM or power savings at the cost of taking more of one or both of the other resources. For example, the fixed keys in each node: in sTiki, each node has a master key from which two keys are derived, in order to secure the communications with the key server. One is for encryption purposes, the other one is used to compute the MACs. It is possible to either store the master key in ROM and derive the two keys on startup, or precompute and store them in ROM so that the power to compute them can be saved on the expense of storing double the amount of keys. If the node has very little ROM left, it will probably be decided to spend the energy to recompute the keys.
7.5.2. Implementation of sTiki’s components

As described above, the sTiki protocol requires a special implementation for nodes representing the collection point of data using TinyIPFIX as their payload, sending data to the gateway for further processing. This node specific implementation is presented below followed by the key server specific implementation. It is responsible for the key creation and the handling of the deployment in order to ensure a secure data connection from node to gateway, respecting the encryption properties.

7.5.2.1. sTiki’s node implementation

For the node implementation part, the main code is integrated in a file called stiki.h using four methods. During startup, the init_stiki() method and the set_stiki_packet_handler(packet_handler ph) method are called. With these calls, the Handshake module resets the handshake status to IDLE, derives the keys for communicating with the server from the initial master key IMK and registers the connection the the key server S. Further, it is determined what should be done with incoming data, meaning which module they should be forwarded to. The stiki_udp_sendto(...) method encrypts the data before sending it, otherwise for unencrypted communication, the stiki_udp_send_to(...) method is selected. stiki_receive(...) is the function to be registered as a callback when setting up connections that will be used to send data.

The stiki-main.c and stiki-main.h files represent the core implementation of sTiki. Any action (e.g. sending, receiving) needs to pass through it. These two methods function as a dispatcher to the correct submodule (e.g. Datatransport module or Alert module). A receiving device automatically directs the packet received to the correct submodule. In case of a sender, a check for existing session or potential errors (e.g. lost IV or keying material) is required first before directing the packet to the corresponding submodule. In the two files structs and helper functions are also included for the configuration purposes of sTiki, such as encryption key length, nonce size, handshake timeout, and number of active sessions.

The designed handshake is included in the Handshake module. This module is responsible for initiating a new handshake and responding to handshake packets received. The initiate_handshake(...) and handle_handshake(...) functions are responsible for both. In order to track the status of an ongoing handshake, the stiki_handshake_context struct is used, holding the following information (Siffert 2018; Schmitt 2019):

- In state the process is tracked to ensure that no messages are skipped during the handshaking and to determine if a new handshake can be initiated, meaning no other handshake is running in parallel.
- target_id stores the identification (ID) of the handshake partner.
- partner_comm stores the (UDP) connection to the handshake partner.
- **target_ip** stores the IP address of the handshake partner, which is included in every message to the partner.
- **my_nonce** stores the current handshake’s nonce, avoiding irritation to previous handshakes.
- **my_token** stores the secret part of the last handshake message.
- **timestamp** stores the corresponding timestamp for the last activity in the ongoing handshake.
- **stored_msg** and **stored_msg_len** stores the message and its length that was sent for handshake initiation.

When a handshake is queried to start, the **initiate_handshake** function is called from **stiki-main**. As defined in section 7.5.1, only one handshake is allowed at a time. Thus, the context attribute **state** is checked to see if it states IDLE and the **timestamp** attribute is checked for a timeout. If all of the checks are passed, there is no ongoing handshake, all context attributes are reset, and a new handshake is initiated by sending message $M_1$ and filling the context attributes accordingly. The **context** is filled as soon as handshake packets are received, which in turn feeds the message to the appropriate handler, creating messages $M_1$, $M_3$, and $M_4$. For message $M_2$ no handler exists, as this message should always be sent to key server $S$. As soon as the handshake is completed, the context attribute **state** is updated and the established session is added to the **Sessionstore**, where the session keys, IVs, and MCs are stored.

The **Datatransport module** is in charge of handling the data received or ready for sending using sTiki. The **datatransport_sendto(...)** method is activated when data should be sent, assuming that a valid session already exists. The corresponding session is fetched, and a corresponding packet including an IV (assuming it was already sent) is created and sent to the destination node. If a packet is received, the **handle_data_transport(...)** method is activated. It is checked if a valid session exists. If not, an alert (via the **Alert module**) is triggered and sent to key server $S$. In return $S$ informs the sender of the missing session. In case the session exists, but the IV is missing, an alert (via the **Alert module**) is also triggered this time, causing a message to be sent to the sender requesting the IV in the next message. In case no alerts are triggered, everything is as required. The message is decrypted and the content is forwarded to the application (i.e. TinyIPFIX) using sTiki.

The **Sessionstore module** receives an entry as soon as two nodes successfully complete a handshake. The module stores a predefined amount of sessions and offers ways to access and manipulate them. The amount of valid session is set in a compiler flag determining the maximum number. An array called **sessions** stores pointers to the sessions, allowing interactions with specific sessions. Some examples include searching for a session, checking if an active session to a specific node exists, updating timestamps for a session, or creating a session. Sessions can also be deleted,
for example when the message counter rolls over, which forces a new session to be established because key or IV renegotiation logic was skipped in favour of a smaller implementation. A session requires 80 B storage and includes the following information (Siffert 2018; Schmitt 2019):

- node_id stores the partner node’s ID.
- my_iv stores the IV used for messages sent to the partner node.
- remote_iv stores the IV received from the partner node.
- crypt_key stores the encryption key for messaged.
- mac_key stores the key used for MAC computation.
- last_event stores the last message’s timestamp between the two nodes, which is used to determine a session timeout.

The functions required for performing encryptions are integrated in the Encryption module. In the current implementation AES-128 is used, supported by the current version of Contiki. The encrypt_message function is responsible for encryption and decryption and requires a given key and the IV. The Encryption module itself applies bitwise XOR to the message for encryption. For decryption the bitwise XOR is applied again, delivering the original input per definition. The encrypt_message function also performs the integrity check, computing and verifying the MAC of a message according to Song et al. (2006). Additionally, this function is responsible for generating the random data used to generate nonces and IVs.

7.5.2.2. sTiki’s key server implementation

The key server’s implementation, consisting of a main dispatcher module and modules for each subprotocol (encryption and session handling), is programmed in Java and integrated into CoMaDa. Here, packet processing is done by stacking protocols on top of each other. Concretely, this means that when a packet is received it is parsed to the protocol processing function and fed into the first protocol (here sTiki, see Listing 7.1). The output is used as the input for the next protocol and so on, until the last protocol has done its job. A packet is not forwarded to the next protocol if the current protocol returns null. An example is a control message. As line 20 shows, processing in sTiki is done by using a function called process( ). It manipulates session information and packet content according to the sTiki protocol and returns a new packet with decrypted content if it was a “datatransport packet”. In order to select the correct processing function, the correct subclass of STikiPacket( ) (see line 9) is determined by looking at the packet header when parsing the packet. Depending on the selection, the required manipulations are performed. For example, in case of handshake and alert packets, the session information gets updated. In case of datatransport packets, the content is additionally decrypted and returned in a new packet.
public WSNProtocolPacket process(WSNProtocolPacket packet) throws WSNProtocolException {
    if (!isSTikiPacket(packet)) {
        return packet; // not an sTiki packet, do not touch
    }

    STikiPacket tikiPacket = STikiPacket.parse(packet);
    if (tikiPacket.missesSession()) {
        AlertPacket.sendInvalidSession(tikiPacket,
                getSourceId(), 1/*Key server ID*/);
        return null;
    }

    if (!tikiPacket.macIsValid()) {
        return null;
    }

    return tikiPacket.process();
}

Listing 7.1. sTiki’s main processing function on CoMaDa (Siffert 2018)

In order to have an efficient implementation in place the code was analysed concerning repetitions. It was recognized that this was the case for cryptographic functions and, thus, it was decided to locate the required classes in one package, called Crypto. It now includes the following classes: (1) AES.java, responsible for encrypting a single block with AES or an entire message in counter mode; (2) AES_CMAC.java, computing the checksum of messages or checking the MAC for correctness; (3) CryptoUtils.java, containing functions to generate random data, converting byte arrays to strings and the other way around; (4) KeyStore.java, interacting with the file containing the initial keying material; (5) STikiUtils.java, containing functions converting IP addresses to node IDs and backwards, logging data and manipulating data (i.e. message header).

It is essential to complete the server side’s implementation functions for storing nodes’ IMKs, process the handshake message $M_2$ and relay the missing_session message to the right node, in order to initiate a new handshake.

7.6. sTiki’s evaluation

As sTiki will run on constrained devices, special focus in the implementation was on resource usage, especially memory and energy consumption. The node’s implementation requires an additional 4,538 B of ROM and 368 B of RAM for collectors, compared to the already existing TinyIPFIX implementation. For the
aggregator device, an additional 4,556 B ROM and 368 B of RAM are required, compared to TinyIPFIX. The overhead is only caused by sTiki, the operating system’s configuration stays the same as before. With this memory consumption measured, sTiki requires the OpenMote platform as a minimum, meaning it cannot run on TelosB. Thus, if a heterogenous network is in place, only parts of the communication can be secured with sTiki. Compared to TinySAM, sTiki is more memory efficient as only one handshake is allowed at a time. Savings in memory are also gained compared to MiniSec, but here especially in the time for encryptions, since MiniSec uses the Offset CodeBook mode to compute the ciphertext and the MAC in a single pass. Energy consumption measurements show that nodes in the standby-mode consume 29.5 mA ± 1.0 mA. When nodes send messages the current increases to 31.5 mA ± 0.2 mA. This was measured independent of the sTiki support as expected, because sending and computing data is the same process in both cases. Only the duration to perform the processes differed, due to MAC calculation, header creation, and session management.

7.6.1. Secured communication between aggregator and server

As a first step in sTiki, the aggregator with ID AEFD establishes a session with the key server that is at the same time the sink in the deployed network. Due to this special situation for the key server, the messages \( M_2 \) and \( M_3 \) of the handshake can happen internally at the sink and only the messages \( M_1 \) and \( M_4 \) are required.

In order to initiate the handshake of sTiki, the aggregator (ID AEFD) sends message \( M_1 \) to the key server. As in this case, the aggregator is the initiator it matches, “Node A” in the protocol’s description. The captured packet is shown in Figure 7.6, where the highlighted 10 byte represent sTiki’s \( M_1 \) content. The bytes marked in red represent the header of sTiki including the magic number (here: \( EF \)), showing that the packet is in sTiki format and the respective protocol information (here: 41). The next four bytes marked in blue represent the node’s ID (here: \( AEFD \)) and the destination’s ID (here: key server with ID 0001). The last four bytes marked in pink
build the random nonce, which will be sent back to the initiator node to determine which request the response belongs to.

![Key Server Sink Aggregator Collector](image)

**Figure 7.7.** Testing setup for sTiki (Siffert 2018)

As assumed due to the setup shown in Figure 7.7, the sink linked to the key server checks if the received message $M_1$ is a valid request. Listing 7.2 shows a debugging output captured in CoMaDa when message $M_1$ is received and the check was passed successfully. Lines 2–4 are responsible for the resolution of the received message $M_1$. In line 5, the MAC validation is done, which will always be successful as $M_1$ is not protected with a MAC. The token (see line 7) is the most important part of the response, because it consists of the following session details (Siffert 2018; Schmitt 2019):

- the involved nodes (here AEFD and 0001);
- following the nonce for matching, the response to the handshake’s initiation;
- the last 16 B building the session key.

Line 6 shows the random IV, which is required together with the node’s IMK to encrypt the entire token. In order to prove later that the packet has not been tampered with, a MAC is added to the message in line 9. Here, the MAC is calculated using the node’s IMK. When everything is in place, message $M_4$ can be sent out by the sink as a response to the initiating node. Figure 7.8 shows the captured message in Wireshark, where the header is marked red (here 44 shows that it is handshake packet $M_4$), the IV is marked blue, the token is marked pink, and the MAC is marked green, resulting in a total length of 57 B.
When the handshake is completed, the aggregator can send data. As no data was sent before, the first message includes an IV. Such a data message is illustrated in Figure 7.9. Here, a message is shown, including an IV that is obvious as the header is $EF_{21}$ (marked in red). The upcoming 13 bytes describe the IV and are marked blue, followed by the data itself. The last 16 bytes of the message marked in pink build the MAC. When this packet is received, the just-received IV and the session key are used to decrypt the data packet. Once an IV was set, the upcoming messages include a MAC (marked in blue) only, saving 11 bytes per message. Such messages are than indicated with a header $EF_{22}$ (marked in red), as shown in Figure 7.10. Again, the MAC is marked in pink and represented by the last 16 bytes in the packet. Listing 7.3 illustrates the respective debugging output from CoMaDa. In line 3, the header details...
are checked, followed by a check to see if an active link exists to the source node (lines 4–5) and if the MAC is valid (line 6). If all of the checks are fine, the data is decrypted and uploaded (lines 7–9).

**Figure 7.10.** Data Packet with a MC captured with Wireshark (Siffert 2018). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip

```plaintext
Listing 7.3. Debugging output for Data Packet received with MC (Siffert 2018)
```

**Figure 7.11.** sTiki message $M_2$ captured with Wireshark (Siffert 2018). For a color version of this figure, see www.iste.co.uk/khatoun/cybersecurity.zip
7.6.2. Secured communication between collector and aggregator

Communications between the collector and aggregator work similarly as described above with one difference, none of the participating nodes are a key server. Here, $M_1$ is sent from the collector to the aggregator and the aggregator sends message $M_2$ to the key server. Figure 7.11 illustrates the corresponding capturing of $M_2$, where the header is marked in pink, the node IDs (here $BFBF$ and $AEFD$) are marked in blue, and nonces are marked in pink. Based on the header (here: EF42), the key server is able to recognize that the received message is $M_2$ of sTiki’s handshake and checks if the nodes are allowed to talk. This check is positive when the key server holds the IMK of each node, because per assumption they count to the same network. If the check is positive, the key server chooses a session key and calculates tokens ($T_A$ and $T_B$) for both nodes, which include the IDs of the nodes ($BFBF$ and $AEFD$), respective nonces ($3BE751AA$ and $2159DB9A$), and the session key. Each token is encrypted with an IV and the node’s IMK. Finally, the key server computes a MAC for $M_4$, because node B is not aware of node A’s IMK. Putting everything together as described in the sTiki protocol results in message $M_3$ (see Figure 7.12, red = header, blue = IV, green = $T_A$, yellow = MAC for $M_4$, orange = $T_B$, pink = MAC), which is sent to node B, forwarding the token to node A, which in return can start with sending data.

sTiki also supports error recovery. The implemented recovery mechanism is activated as soon as (1) an IV or (2) a session is lost. Both result in sending alert messages in sTiki. Debugging examples are shown in Siffert (2018). For case (1) it is important to mention here that the not decryptable packet is lost, because the last received messages are not stored in the memory, and an alert packet is generated to send to the communication partner, indicating that the IV is missing, and including the following information: (a) the header $EF63$, (b) the IDs of the involved nodes (e.g. $AEFD$ for the aggregator and 0001 for the key server), and (c) the MAC computed with the session key, because an active session between the two nodes exists. For case (2) the receiving node would want to try to inform the sending node about the
problem. As they do not have a shared session, the receiving node cannot do this in a secure manner. If the notification is possible in an unsecured manner, like sending a unsecured packet to initiate a new handshake, a DoS attack becomes possible by sending missing session packets to the complete network continuously. Thus, the message is relayed by the key server that can notify the sender node using its IMK to initiate a new handshake. In such a case, the key server recognizes the message as its header equals $EF65$. The packet includes the IDs of the involved nodes (e.g. $BFBF$ and $AEFD$) protected by a MAC, which is computed with the IMK of the sender node. The key server constructs a message with header $EF65$, including the two node IDs to the destination node. The message is protected by the IMK of the destination node. The destination node is able to decrypt the packet. Based on the included information, the destination node initiates a new handshake with the respective node.

<table>
<thead>
<tr>
<th></th>
<th>Collector ROM</th>
<th>Collector RAM</th>
<th>Aggregator ROM</th>
<th>Aggregator RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System Contiki, TinyIPFIX</td>
<td>48,248</td>
<td>15,999</td>
<td>48,357</td>
<td>17,073</td>
</tr>
<tr>
<td>Operating System Contiki, TinyIPFIX, sTiki</td>
<td>52,796</td>
<td>16,367</td>
<td>52,913</td>
<td>17,441</td>
</tr>
<tr>
<td>sTiki only</td>
<td>4,548</td>
<td>368</td>
<td>4,556</td>
<td>368</td>
</tr>
</tbody>
</table>

Table 7.1. sTiki size according to size utility [Byte]

### 7.6.3. Communication costs

Power consumption was measured by connecting a multimeter in between one side of a battery and the power socket on the node. Multiple attempts were made and produced very similar results. During standby, the nodes had a power consumption of 29.5 mA ± 0.1 mA. When sending messages, the current increased to 31.5 mA ± 0.2 mA. This was the same on nodes with and without sTiki, which is to be expected because sending and computing data is the same process in both versions. The durations of those phases should be longer on the version with sTiki because the messages have some overhead because of the header and MAC, and the calculations should also take longer because the version with sTiki has to compute the MAC and manage sessions. On the nodes with sTiki the multimeter occasionally showed a lower current, flowing around 28 mA. It is unclear where this comes from, one possibility is that the hardware implementation of AES takes less power and the processor pauses calculations while the encryption is going on. However, the drops are not frequent enough to match with every encryption operation, but this may be caused by a low sampling rate of the multimeter, which misses short encryption sequences.

By analyzing the binary file generated by the `make` command with the utility `size`, it is possible to find out the ROM and RAM requirements of the code running on the nodes. Table 7.1 shows the measurements. It shows a RAM usage of 368 B (reported as `text`) and a ROM usage of 4,548 and 4,556 B (reported as `data` and `bss`) for sTiki.
This difference in ROM usage most likely results from differing amounts of code that calls sTiki and/or differing compiler optimizations.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <match>
    <field>
      <name>Temperature (Sensiron SHT11)</name>
      <fieldID>0x80A0</fieldID>
      <enterpriseNumber>0xF0AA00AA</enterpriseNumber>
      <type>Temperature</type>
      <unit>Celsius</unit>
      <expression><![CDATA[ Math.round((−39.7+(0.01*x))*100)/100 ]]> </expression>
    </field>
    <field>
      <name>Humidity (Sensiron SHT11)</name>
      <fieldID>0x80A1</fieldID>
      <enterpriseNumber>0xF0AA00AA</enterpriseNumber>
      <type>Humidity</type>
      <unit>%</unit>
      <expression><![CDATA[ Math.round((−2.0468+(0.0367*x) + ((−0.0000015955 * (x * x)) * 100) / 100)]]> </expression>
    </field>
    ...
  </match>
</configuration>
```

Listing 7.4. XML example for sensors (Sgier 2017)

### 7.6.4. Integration into an existing system

In order to configure and manage the deployed network using Contiki, the server component shown in Figure 7.3 was extended by the Contiki support including a user-friendly Graphical User Interface (GUI), called CoMaDa, offering Configuration, Management, and Data handling functionality for the deployed network. The GUI works according to the “click buttons” principle. CoMaDa itself offers a panel to view received data after the network was deployed, the tunnel was activated, and the virtual network interface was created. In order to decode the received packets correctly, CoMaDa requires an XML (eXtended Marup Language) file, where all parameters for each sensor are stored together with the mathematical equation to calculate the correct value. An example is shown in Listing 7.4. A “field tag” is specified for each sensor of the device (e.g. lines 4–12 for a temperature sensor, lines 13–21 for a humidity sensor). The field tag for pull support is already included, but not yet
activated in the implementation. Each enclosing field tag represents a Template Record from TinyIPFIX with additional information (Sgier 2017; Schmitt 2019):

1) **name** states the sensors name and indicates the sensor vendor (e.g. temperature (Sensiron SHT11) – line 5).

2) The **fieldID** and the **enterpriseNumber** are used to identify the data source uniquely (e.g. sensor from vendor X – lines 6–7).

3) **type** defines the type of the value measured (e.g. temperature – line 8).

4) **unit** defines the respective unit of the measured value (e.g. Celcius – line 9).

5) **expression** includes the mathematical equation translating the bit string received into a meaningful value, consistent with the **type** and **unit** information (e.g. lines 10–11)

As a result, the person sitting in-front of a device running CoMaDa can see the received data in a live stream, as shown in Figure 7.13. The data is sent to WebMaDa’s backend to store it in WebMaDa-DB and make it accessible via WebMaDa’s front-end, as described in Schmitt (2019) for a live-stream with \( doa = 1 \), meaning no aggregation is performed.

As can be seen in Figure 7.13, it is not visible in GUI CoMaDa if encrypted communication was activated or not. Thus, a proof of operability is performed by showing the captured messages in Wireshark, assuming the network setting shown in Figure 7.7 and that the aggregator performs message aggregation with \( doa = 2 \). Here, the aggregator waits until two messages are received from the collector with ID BFBF before it performs aggregation.

### 7.6.5. Comparison to existing approaches

The implementation of TinySAM on TinyOS in (Lowack 2013) uses almost 6.5 kB of ROM and about 1.5 kB of RAM with similar configurations, which makes sTiki 2 kB smaller in ROM and 1.1 kB smaller in RAM usage. The smaller size is mainly due to the limit of only one ongoing handshake at a time. Further reasons seem to point to the AES implementation, but such an evaluation was outside the scope of this work.

In comparison to TinyDTLS, sTiki requires about 15 times less ROM and 50 times less RAM. This large difference was expected, because DTLS requires three different encryption algorithms. Just the binding to RSA takes about the same amount of ROM as the entirety of sTiki (Kothmayr et al. 2013). The price for using such a small implementation is its non-compliance with the standard DTLS, which might be more important in some use cases.
sTiki’s strength compared to MiniSec is size: it uses about half as much RAM and a little less than four times as much ROM as MiniSec uses. The important tradeoff during operation in comparison to MiniSec is in encryption speed, since MiniSec uses an Offset CodeBook mode to compute a ciphertext and MAC in a single pass. sTiki uses a separate algorithm for those two operations, thereby spending more time and energy to compute.

Comparing the power consumption in detail with these measurements collected is not feasible, since precise timings are not available for the Contiki implementation.

![Figure 7.13. CoMaDa’s live stream showing the message aggregation](image)

### 7.7. Summary and conclusions

This book chapter detailed insights about the measurable paradigm change of the initial Internet toward the IoT. Due to this situation and the rising awareness of personal data’s misuse, security concerns have been identified. As it was depicted, reasons for delivering these security measures in the IoT occur due to (a) constrained device characteristics and (b) communication protocols used (e.g. IEEE 802.15.4
or ZigBee). Thus, effective and efficient security protocols are demanded for IoT scenarios.

`sTiki` was designed as such a security protocol to face the aforementioned challenges to overcome the security concerns mentioned. `sTiki` was designed in a modular and efficient manner, mapping the resources of constrained devices to possible functionality. As seen in these evaluations, secure communication can be established in a constrained network, however, it remains invisible for the user, when only viewing data received in graphical user interfaces, such as CoMaDa and WebMaDa. A key server implementation is resource consuming and not intended to be implemented on constrained devices itself, but with `sTiki` the key server is implemented on a resource-full instance of CoMaDa, following the delegation manner and, thus, management work is performed there instead of within the nodes.

`sTiki` strengthens the constrained network deployed in its security: authentication and data integrity support is due to MAC usage, freshness support reached by counters and confidentiality via encryption using individual link-based session keys. Currently, symmetric encryption is used, but with a modification asymmetric encryption is feasible, too, if devices offer sufficient resources. Additionally, `sTiki` supports error recovery that can occur, (a) if a node has lost the initialization vector or has detected a packet with the wrong MC or (b) if a node has lost keying material, especially the session key. In the case of (a) the node causes the communication partner to send a new IV with the next message by sending a respective message. In case (b) the node will relay a command to have the other node initiate a new handshake via the key server. This message is sent via the key server, because the key server is the only node that operates with a key to communicate securely with the other node.

Overall, this chapter demonstrated that efficient security can be implemented in the IoT on constrained devices for smart home applications, collecting periodic data that addresses many security concerns from users. Furthermore, with `sTiki`’s implementation and assuming OpenMote B as the node’s platform, sufficient memory stays in place for further IoT applications on such constrained devices, such as data aggregation or collection. Due to the delegation of the main security operation of creating key material, the energy consumption for encrypting messages in the network is highly viable for IoT scenarios. In order to validate `sTiki` for industrial purposes, it is recommended to use an automated validation tool, such as Automated Validation of Internet Security Protocols and Applications (AVISPA).

7.8. Acknowledgements

The work was supported partially by the University of Zürich UZH, Switzerland, and the European Union’s Horizon 2020 research and innovation program’s project Concordia, under Grant Agreement No. 830927.
7.9. References


Sgier, L. (2016). Optimization of TinyIPFIX implementation in Contiki and realtime visualization of data. Software project, Communication Systems Group, Department of Informatics, University of Zurich, Switzerland.

Sgier, L. (2017). TinyIPFIX aggregation in Contiki. Internship, University of Zurich, Switzerland.


Tex (2012). CC2538 a powerful system-on-chip for 2.4-GHz IEEE 802.15.4-2006 and ZigBee applications [Online]. Available at: http://www.ti.com/product/CC2538 [Accessed 14 August 2018].


List of Authors

Mirna ATIEH
Computer Science Department
Lebanese University
Beirut
Lebanon

Fayez GEBALI
Department of Electrical and
Computer Engineering
University of Victoria
Canada

Rida KHATOUN
Télécom ParisTech
Paris
France

Adrian KOTELBA
VTT Technical Research Centre of Finland Ltd
Espoo
Finland

Vikas KUMAR
Central University of Haryana
Mahendergarh
India

Manju LATA
Chaudhary Bansi Lal University
Bhiwani
India

Axel LEGAY
UCLouvain
Ottignies-Louvain-la-Neuve
Belgium

Jean Pierre LORRÉ
Linagora Grand Sud Ouest
Toulouse
France

Mohammad MAMUN
National Research Council of Canada
Government of Canada
Canada

Melody MOH
Department of Computer Science
San Jose State University
USA
Index

A, B
anomaly detection, 204, 223, 233–237
artificial intelligence (AI), 193, 207, 221, 222, 224, 225, 227, 228
authentication, 15, 21, 49, 75, 125–132, 151, 172, 178, 185, 189, 214–220, 245
device, 107, 111, 125, 127, 129, 151, 214
mutual, 75, 128, 132, 246
Bluetooth, 2, 10, 25, 104, 160, 165, 170

C, D
challenge/response pairs (CRP), 127–131, 142, 144, 150
CMOS noise, 125, 128, 133, 138, 140
constrained networks, 9, 251–253, 280
conversational application, 197
data privacy, 178, 179, 187, 191, 194, 197,
datagram transport layer security (DTLS), 11, 85, 86, 90, 99–101, 113, 171, 247, 263, 278
deep learning (DL), 126, 128, 150, 203

H, I
HAIFA, 3–5, 12, 14–32, 35–48, 54–56, 62–65, 68, 69
handshake, 97, 100, 101, 171, 261–273, 275, 276, 278, 280
human activity recognition (HAR), 227
Internet of Things (IoT), 155, 177, 203
network security, 155
security, 87

L, M
LinTO, 194–197
machine learning (ML), 78, 91, 150, 162, 204, 209, 221–228, 231, 237
message queuing telemetry transport (MQTT), 4, 10–12, 14–16, 25–27, 32, 48–55, 86, 90, 180, 195

N, O
near field communication (NFC), 165, 167, 215
neural network (NN), 204, 221, 224, 225, 228–232, 237
Open Connectivity Foundation (OCF), 160, 169
OpenMote B, 68, 257, 280

P, S
Pi Platform, 177, 179, 190–192, 197
secret session key, 105, 150
SecureWSN (wireless sensor network), 1–5, 7, 12, 16, 18, 25, 29, 64, 68, 69

© ISTE Ltd 2022. Published by ISTE Ltd and John Wiley & Sons, Inc.
security
  home, 155, 157, 160–162, 169
  standards and regulations, 155, 156, 160, 161, 169, 171, 173
SRAM PUF, 125–151

U, Z
  unique device identity, 127
  zero-trust network, 177–199
ZigBee Alliance, 9, 165, 170, 171