Situation-aware Adaptive Cryptography

VIKTOR HJELM
MARC TRUEDSSON
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Situation-aware Adaptive Cryptography

Viktor Hjelm
bas12vhj@student.lu.se

Marc Truedsson
elt13mtr@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Sebastian Mauritsson (Advenica), Niklas Lindskog (Advenica), Paul Stankovski (EIT)

Examiner: Thomas Johansson

February 28, 2018

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The matter of security is often a trade-off between protection and performance. The fact that more and more small, battery powered and computationally weak devices are connected to the Internet, makes this matter more relevant than ever. Such devices could be smart-watches, smart smoke detectors, IP Cameras, etc. While some of these devices completely lack data protection, other devices have implemented classic security schemes which are not optimized for such devices.

This thesis investigates the concept of situation-aware adaptive cryptography in an IP Camera, where different security schemes can be used depending on the current situation of a device. A number of different security schemes are implemented, tested and compared to each other. From this, different protection levels are proposed not only containing classic cryptographic functions but also lightweight cryptographic schemes specialized for this new type of small devices.

Different parameter measurements, relevant for assessing the situation of a device, are also implemented and compared to each other in terms of power consumption, time, and data sent. Finally, the results from these two studies are put together into a complete concept solution, implementing the full situation-aware adaptive cryptography scheme. This particular implementation can save up to 53% of the power used for encryption and sending of data, in an average use case about 26% is saved.

Keywords - Situation-aware Cryptography, Adaptive Cryptography, Lightweight Cryptography, IoT, Intrusion detection, Energy efficiency

Acknowledgements

We would like to thank Advenica and its people for being very friendly and accommodating throughout our thesis. Special thanks go out to our supervisors Sebastian Mauritsson and Niklas Lindskog for their valuable support, feedback and positivity. Thanks to Henrik Normann, Tobias Claesson and Oskar Jönsson for their input regarding this thesis.
We would also like to thank our supervisor from LTH, Paul Stankovski, for his support and academical points of view during this thesis.

Table of Contents

List of Figures

List of Tables

Abbreviations

AES	-	Advanced Encryption Standard
ARP	-	Address Resolution Protocol
DHCP	-	Dynamic Host Configuration Protocol
DNS	-	Domain Name System
FTP	-	File Transfer Protocol
HMAC	-	Hash-based Message Authentication Code
HTTP	-	Hypertext Transfer Protocol
HTTPS	-	Hypertext Transfer Protocol Secure
IANA	-	Internet Assigned Numbers Authority
ICMP	-	Internet Control Message Protocol
IGMP	-	Internet Group Management Protocol
IoT	-	Internet of Things
IP	-	Internet Protocol
MAC	-	Message Authentication Code
MAC address	-	Media Access Control address
MitM	-	Man-in-the-Middle
NIC	-	Network Interface Card
NIST	-	U.S National Institute of Standards and Technology
NSA	-	National Security Agency
OS	-	Operating System
S-Box	-	Substitution Box
SDP	-	Service Discovery Protocol
SHA	-	Secure Hash Algorithm
SMTP	-	Simple Mail Transfer Protocol
SPN	-	Substitution-Permutation Network
SSH	-	Secure Shell
TCP	-	Transmission Control Protocol
TLS	-	Transport Layer Security

Chapter 1
Introduction

This thesis will investigate and research the concept of situation-aware cryptography, for the sake of Advenica AB. All research has been conducted at Advenica AB in association with the Electrical and Information Technology department at the Faculty of Engineering, Lund University.

In this chapter, an introduction to the thesis will be presented, consisting of the background to the research, project aims and questions to research, the scope of this thesis, contributions from the different authors and the outline of the thesis. Finally, the methods used to create, test, and measure a situation-aware adaptive cryptography scheme will be presented.

1.1 Background

Today the number of devices connected to the Internet, and to each other, is increasing faster than ever. What separates many of these devices from classic computers or servers, is that they come in all different forms and are used in all kinds of areas. A common denominator for many of these devices is that they have a need for some type of security. Some of these products handle private or user-sensitive data, e.g fitness bracelets or IP cameras. Since devices like these often run on a battery, it is of great essence that they are very power efficient. Given this requirement comes a trade-off between security and power efficiency [52].

Classic security methods, such as TLS, may no longer be the most suitable option for such devices because of these hard constraints. Instead, new ways of achieving confidentiality and integrity protection, using less energy and computational power, have been proposed. This type of algorithms, often called lightweight-cryptographic algorithms, can be less secure than classic cryptographic algorithms due to shorter key-sizes or not as well-studied algorithms. One approach to handle this problem is to use a mix of different cryptographic algorithms and adapt the cryptographic algorithms to the situation. With this approach, a more secure cryptographic algorithm can be used in a situation where an attacker is more likely to be present and the data sent is particularly sensitive to the user. In another scenario, where the data is less valuable to the user, and an attacker is less likely

to be present, the data can be sent using a lightweight-cryptographic algorithm, and therefore save both computational power and energy for the device.

1.2 Project Aims

The objective of this project is to research the possibilities to create a situation-aware adaptive cryptographic scheme, that can be used for devices with limited processing power and energy consumption. In the case of this project, the considered device is a battery powered IP camera. A possible way of the information flow in a situation-aware IP Camera is shown in figure 1.1. The camera is for private use and is assumed to be in an indoor environment. The attacker is assumed to have access to the same local network, with the ability to read and alter the data transmitted. The camera is to protect data in such a way that the attacker is not able to read or alter sensitive information. The idea is to use existing cryptographic schemes with various protection levels and processing power, depending on the situation the device is in. The device has to determine and choose the scheme most appropriate for the situation.

1.2.1 Questions to research

1. How should a device monitor its network communication to determine a threat level? What should it react to?
2. How often should the device evaluate and adapt to the current threat level?
3. Under what circumstances would it be possible and useful to implement adaptive handling of security protocols in a device with limited processing power?

1.3 Scope

The aim of this paper is to investigate the usefulness of situation-aware cryptography on a concept level, as well as proposing a way of creating such scheme. In practice, it is possible to create adaptive cryptography in many different ways. There are a lot of different cryptographic algorithms which would be suitable for such a scheme. The purpose of this paper is not to find optimal cryptographic algorithms for a particular use case. Instead, the focus is a proof of concept solution for an efficient way of using different encryption and hash algorithms, to achieve adaptive cryptography.

In a similar manner, the proposed solution does not aim for perfectly mapping the measured situation to a protection level. In order to do so, more information about what kind of data that would be sent is required, as well as more detailed information about what different threats to consider for that particular use case. Instead, the authors of this paper leave those decisions for the user of such a situation-aware solution.

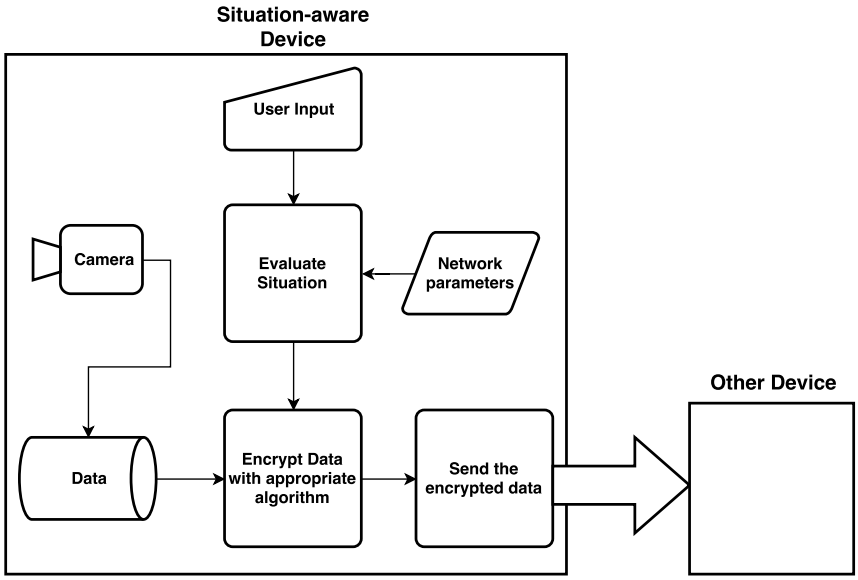

Figure 1.1: Flow of the Situation-aware Device.

This paper won't consider any authentication or key establishment issues. Instead, a pre-shared key is assumed for all use cases. In reality, this may not always be the case, but due to the limited time of this thesis, and the fact that there exist many different approaches to such problems, they have been left out of this project.

1.4 Contributions

During the work of this thesis, the tasks have been distributed evenly among both participants. Both participants have been equally involved in all parts of the thesis project. Since both authors share a lot of knowledge from the same courses and Master specialization there have not been any problems in working closely together with all tasks. During the writing of this thesis, both authors have been involved in every part.

1.5 Outline

The outline of the remainder of this thesis is as follows:
Chapter 2, Threats: Describes the attacks and threats handled in this thesis.
Chapter 3, Cryptographic Algorithms: This chapter introduces the cryptographic algorithms which are used in order to provide confidentiality and integrity depending on the protection level.
Chapter 4, Situation Parameters: The different parameters, which are taken into consideration when determining the threat level, are described.
Chapter 5, Adaptation and Evaluation: Evaluation of the parameters from

the previous chapter and mapping them to the Protection levels.
Chapter 6, Throughput Tests: Throughput result of the cryptographic algorithms is presented.
Chapter 7, Power Consumption Tests for Protection Levels: Results of tests performed to measure the power consumption of the different cryptographic algorithms and in turn the protection levels.
Chapter 8, Situation Measurements: The power consumption of the situation parameters is tested. The time for each measurement is also recorded.
Chapter 9, Protection and Situation-awareness Comparison: The cost of the situation measurements is compared to the cost of encryption in order to determine the usefulness of the implemented solution.
Chapter 10, Conclusion: A conclusion of the work done and the result of this thesis. The conclusion also contains suggestions for future work.

1.6 Method

This thesis will develop and implement a situation-aware adaptive cryptographic scheme, which will be the basis of a number of simulations and tests meant to answer the questions to be researched from Section 1.2.1. The proposed solution consists of three main parts, Situation parameters, Adaptation, and Protection, to achieve situation-aware adaptive cryptography. Figure 1.2 shows a block diagram representing this solution.

The first part, **Situation parameters**, will measure relevant network parameters related to Man-in-the-Middle attacks, a malicious user sniffing the network traffic, intrusion detection, and the topology of the network. Apart from the threat related parameters, situation parameters also include the sensitivity of the data. This part provides necessary information for determining the current situation and threat level of the device.

Adaptation handles the evaluation of the input provided by the measurements and the user from the Situation parameters part. The job of the evaluation engine is to process the values from each measurement and assess the current threat level for the device and match this to a protection level. The protection levels could be seen as different alert states, similar to what is used in many other systems today such as DEFCON [20].

Protection handles the different protection levels, which perform the cryptographic algorithms associated with each level. The levels come with a trade-off between security and time- and computational complexity.

The work will be separated into three major parts. The first part is a theoretical study necessary for the development of the proposed scheme. This study includes the different cryptographic algorithms used, the potential threats and vulnerabilities as well as appropriate situation parameters to measure for this project. The second part is a model of the scheme implemented in Python. This model will be

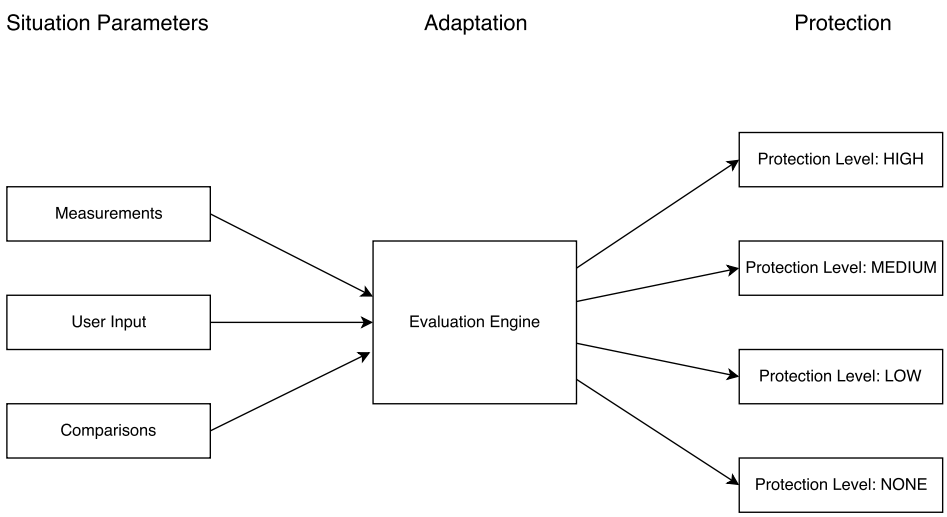

Figure 1.2: Architectural overview.

the basis for tests evaluating throughput between the different algorithms and the time consumption of the different measurements gathered. The results of these tests will be done to confirm if the cryptographic algorithms used, and the measurement gathered, is suitable for the final implementation. Following the results of the tests performed in the previous part, a final implementation will be made. This implementation will then run on a device, where tests regarding the energy efficiency of the model will be performed.

Chapter 2
Threats

Any device connected to a network is exposed to a number of different threats. IP Cameras are no exception from this. In recent attacks, some cameras have been extra targeted due to various design flaws, such as hardcoded passwords [32]. In order to measure the threat level of a device, it is vital to understand what attacks the device is exposed to, and how these attacks work. The most common, and relevant, threats to consider for this thesis project are Man-in-the-Middle, Sniffing, and Intrusion attacks.

2.1 Man-in-the-Middle attacks

A Man-in-the-Middle, MitM, attack is an attack where a malicious party successfully reroutes all connections between two hosts through the attacker. This means that the attacker is able to read and alter all data sent between the two parties, without the knowledge of either of the two communicating parties. Encrypting the data sent between the two parties can make it harder, or even impossible, for a MitM to read the actual information sent. Similarly, a good integrity protection will alert the communicating parties when the data sent has been altered during the transmission. Figure 2.1 shows a simple example of what the connection looks like during a MitM attack. In this case, Alice and Bob represent the communicating parties and Mallory represents the MitM attacker.

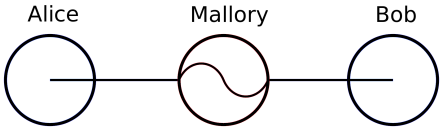

Figure 2.1: Connections during a Man-in-the-Middle attack [58].

The easiest way to picture how a MitM attack is realized is to assume that the attacker is physically connected to the communicating parties. This is not always the case since it could be hard for an attacker to physically alter the connections. Another way to perform the attack is through an ARP cache poisoning [37]. In that case, the communicating parties are tricked into sending data to the attacker on the same local network, instead of directly to each other. In both of these cases, the attacker needs to enable IP packet routing, which allows the attacker to forward IP packages acting as a router [56].

2.2 Network Sniffing

Network sniffing is an attack where the intruder listens to, or monitors, the network communication passively. In this type of attack, there is no altering of data, and there is no need for the intruder to be positioned between the two communicating parties. For these reasons, detecting sniffing attacks is considered hard, although possible in some cases. An overview of the attack is found in Figure 2.2 below.

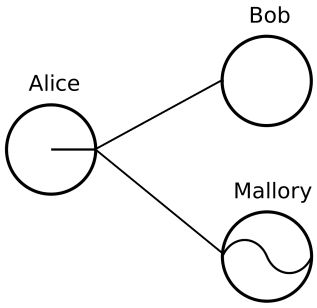

Figure 2.2: Network Sniffing.

In Figure 2.2, Alice and Bob try to communicate and Mallory is the intruder. It seems like Alice is directly sending all information to Mallory and therefore making the attack possible. This is not the case, instead, there exist many possible reasons why Mallory can read the data. For example, if a hub is used, which repeats all received data to all output connections, or that a Wi-Fi is used so all connected devices can listen. Another possible reason could be that the attacker has launched a successful MAC address flooding attack, which forces the switch to enter 'failopen mode' acting as a hub [48].

2.3 Intrusion

A malicious party trying to gain access to a device, it is not allowed to access, is called an intrusion. What differs intrusion from sniffing and MitM attacks, is that the malicious party does not have to break any type of encryption or integrity protection in order to read or alter data after a successful intrusion. Instead exploits in the device software, or configuration errors, could open up for

a successful attack. Since this type of attack is independent of the encryption, this means that changing the protection level of the sent data, which is what is done by the situation-aware adaptive cryptographic solution, is not a direct solution to an intrusion. However, since signs of an intrusion indicate that there is a malicious party on the network, it is still considered a relevant threat to this solution, and if an intrusion is suspected the protection level is to be increased.

Intrusions are often initialized by a port scan to look for services running on the targeted device. The reason for this is to find a running service with a known exploit, and then use that exploit to gain access to or collect information from the target. By listening for incoming connections on ports that are not in use, but typically known for services with exploits, an intrusion attempt can be detected.

Chapter 3
Cryptographic Algorithms

In order to create a successful situation-aware adaptive cryptography scheme, it is of great essence to choose suitable cryptographic functions as well as choosing relevant parameters to react and adapt to. Since the use cases of this project are meant to include data with different degrees of sensitivity, in a variety of different environments, the protection of data should span from very secure, to no security at all.

This chapter will describe the different cryptographic functions used for achieving, not only different levels of security but also different costs in terms of throughput and energy consumption. The cryptographic algorithms handled in this chapter are divided into encryption, providing confidentiality, and message authentication, providing integrity protection.

3.1 Encryption Algorithms

The cryptographic algorithms in this project are meant to reflect different protection levels. When choosing the set of algorithms to use for an adaptive cryptographic scheme it is important to choose algorithms with different levels of security and cost. In this context, the cost is considered to be processing power, throughput, and energy consumption. For a fair comparison, encryption algorithms proposed in this project are all block ciphers with variable key length and block size. It is important to keep in mind that these algorithms are just one proposal, and that similar solutions could use other algorithms or stream ciphers or a single block cipher with a variable number of rounds.

The highest protection level for this project is to use a very secure and well studied algorithm. Traditionally, the power consumption for such algorithms is high, but since it is used only when a threat is assumed to be present, this is to be considered acceptable. The least secure method used in this project is simply to send all information in cleartext. This method, although energy efficient, provides no security and anyone with access to the data can read or alter the information. In between these two levels, there is a need for cryptographic functions which provide less security but with better performance. In the following sections, two different cryptographic functions are presented and compared. The idea for the

cryptographic functions is to not only provide different levels of security, but also different amounts of energy and computational power.

3.1.1 Advanced Encryption Standard

The Advanced Encryption Standard, or AES, is a symmetric block cipher based on a subset of the Rijndael cipher developed by Vincent Rijmen and Joan Daemen. It was adopted by the U.S. government in 2001 as the new encryption standard replacing the Data Encryption Standard, and has since been added to other reputable organization lists of recommendations [19] [41]. AES was designed as a public and flexible block cipher with three different key sizes of 128, 192 and 256 bits. Because of the flexibility in the cipher, as well as the public nature of the source, the use of AES is widespread and therefore a suitable cipher for this project.

The cipher design is a Substitution and Permutation network (SPN), explained more in depth in Appendix A, where the data block to be encrypted goes through this network a number of rounds depending on the key size. AES specifies that 10, 12 and 14 rounds should be used for the key sizes 128, 192, and 256, respectively. This thesis will use AES with a key size of 256 bits since it is the most secure version of AES.

AES is widely used in IP cameras and several other applications today [10] [54]. This is one of the most used encryption algorithms today since it is common for many TLS connections to use AES [22]. AES also fulfills NSA's latest requirements for sharing of TOP SECRET information in national security systems [17]. The negative aspect of this algorithm is that it is computationally heavy, and using it in a device with high energy constraints and low computational power at all times, may not be the best choice.

3.1.2 SIMON and SPECK Lightweight Ciphers

SPECK and SIMON are block ciphers first proposed by NSA in 2013 [6]. They are both lightweight ciphers designed to perform well for constrained devices. What differs the two ciphers is that SIMON is optimized for hardware implementations while SPECK is more suitable for software implementations. This makes SPECK more relevant for the purpose of this paper since the implementations will be done in software only. When proposed in 2013 the two ciphers were said to "outperform both the best comparable hardware algorithms and the best comparable software algorithms" [7].

It is worth noting that both SPECK and SIMON lack S-boxes, a common part of many block ciphers such as AES, which are used for introducing non-linearity. S-boxes are described further in Appendix A. Instead, SPECK uses modular addition for non-linearity. This method is preferred in software over hardware [8]. One thing that makes SPECK particularly well suited for software implementations, is the fact that every operation can be done in-place. This means that there is no extra

memory space needed when performing the operations used in SPECK, which of course is an advantage.

3.1.3 Comparison between AES and SPECK

Table 3.1 shows throughput for SPECK compared to AES. The key size and the block size are 128 bits for both algorithms. The hardware clock speed is 100 kHz and the software clock speed is 16 MHz. Comparing SPECK and AES from Table 3.1 it is clear to see that SPECK, in this case, has a higher throughput than AES when implemented in software, which makes it a suitable algorithm for this project.

Table 3.1: Cryptographic function performance comparison [6].

Algorithm	Throughput Hardware [kbps]	Throughput Software [kbps]
SPECK	12.1	768
AES	56.6	445

At the writing of this thesis, to the best knowledge of the authors of this paper, there exists no full round attacks on the SPECK cipher. Instead, only attacks targeting a reduced number of rounds have been published. In 2013 Abed et al. proposed a way of attacking a SPECK cipher, with a block size of 64 bits and a key size of 128 bits, after 14 out of 27 rounds [1]. The time for recovering the key was $2^{89.4}$, compared to 2^{128}, which is achieved by exhaustive key search, in other words, more than 10^{11} times faster. The best published reduced round attack on SPECK targets about 70% of the total number of rounds done in SPECK, which is considered "A healthy security margin" [9]. This gives a hint about how vulnerable the cipher is considered today.

Full round attacks have been presented on the AES cipher, which in theory means that the cipher is broken. Bogdanov et al. showed in 2011 a way of breaking the 256-bit AES in a computational complexity of $2^{254.4}$ [14]. Comparing this to exhaustive key search which requires a computational complexity of 2^{256}, it is clear that the attack only finds the key about 3 times faster. Similar attacks also exist on the 128-bit version of AES. This makes AES practically impossible to crack using current technology. Putting it in the words of the author Bogdanov: "To put this into perspective: on a trillion machines, that each could test a billion keys per second, it would take more than two billion years to recover an AES-128 key. Because of these huge complexities, the attack has no practical implications on the security of user data." [15].

It is clear to see that both of the cryptographic algorithms have both positive, and negative, aspects. While the 256-bit AES can be viewed as the more secure option, it is still quite costly in terms of computational power and energy for a

variety of devices. SPECK, on the other hand, is very efficient when it comes to energy and computational power. There is also a possibility to choose from a lot of different key and block sizes to adjust the level of security. This makes them both great candidates for an adaptive cryptographic scheme.

3.2 Message Authentication

The protection against tampering of data, along with the authenticating of the sender, is an important feature in a security system. Since the protection levels of a situation-aware scheme may include both message authentication, with or without encryption, it is important to separate these two cases.
Authenticating a message, which is sent encrypted, can be done by hashing the message, and including this hash with the message.
If message authentication is to be used without encryption, a key-less hash is not sufficient to provide this. Instead, a HMAC can be used to provide message authentication. HMAC, which is a hash-based method including a key, can be used with any hash function suggested in this section.
Again, to achieve a useful adaptive cryptography it is important that the different algorithms for message authentication provide different levels of protection and cost, in terms of energy, throughput, and computational power.

This section deals with the different algorithms researched in this project to implement message authentication. Like the case with encryption, the highest level of security reflects well-known and widely used algorithms. These are known as the Secure Hash Algorithm or SHA. In addition, a lightweight cryptographic hash function, which is less costly than the more well-studied SHA functions, is introduced. In similarity with the encryption case, different functions can be used as long as they have different cost and protection level.

3.2.1 SHA2

When encryption is used, there is no need for using keyed hashes such as HMAC for message integrity. Instead, a hash of the data sent can be encrypted by the same algorithm as the message and thus provide the integrity. SHA2 is a hash function designed by NSA and published in 2001. The algorithm is a U.S. standard which is widely used and available in popular cryptographic protocols. The hash family offers output digests of different sizes making it flexible to use. To the best knowledge of the authors, there exists no full round attack on SHA2.

3.2.2 SHA3

SHA3 is the latest addition to the hash algorithms standardized and certified by NIST. Released in 2015 as a subset of a cryptographic family called Keccak which is internally different from the previous algorithms SHA2 and SHA1 [42]. The intention of this new algorithm is not that of a direct substitute to SHA2, but an alternative, widening the options of using a NIST approved hash algorithm.

3.2.3 BLAKE2

One very fast option for hashing is the BLAKE2 algorithms. Its predecessor, BLAKE, was a proposal for SHA3 but got rejected. BLAKE2 is said to be as faster than SHA1 but with security similar to SHA3 and is designed for software implementations [5]. BLAKE2 comes in four different versions, BLAKE2b, BLAKE2s, BLAKE2bp and BLAKE2sp, with each version optimized for usage on different platforms depending on the CPU version.

Another advantage with BLAKE2, apart from the speed, is that it has a keyed version. This means that there is no need for using HMAC, or any similar method, when BLAKE2 is used without any encryption.

Chapter 4
Situation Parameters

Establishing a situation-aware framework, which is supposed to detect whether or not there is a threat on the network, requires some input. This input can be gathered in many ways, where the most readily available input, comes from measuring different network parameters and predefined input from the user. The framework must be able to monitor the network, detect anomalies, and categorize the perceived threat level in order to correctly determine which protection level to implement. Since this thesis is based on the use case of an IP camera, the networks researched are Wi-Fi, which is the main focus, and Bluetooth. The different parameters this project will handle are described in this chapter.

4.1 Wi-Fi

This section details the parameters specific to Wi-Fi in this thesis. Most IP Cameras use Wi-Fi connections exclusively for communicating which makes the Wi-Fi parameters most interesting for the situation evaluation [26].

4.1.1 IP Packet Routing

To investigate if a network is infiltrated by a Man-in-the-Middle, a simple test can be performed to receive possible candidates of such an attack [56]. The test is designed to evaluate whether a device is rerouting data sent in the network, which is the case in a MitM attack. The host, which is the one performing the tests, sends a trap ICMP ping packet to the suspect. The packet is that of an ordinary ICMP ping, addressed to the suspected device. The difference is that the destination IP is not the suspected device, rather the IP of the host doing the testing. If the target host is rerouting packages in the network it will send the ping request back to the test host. This is an indication that the suspect could be deploying a MitM attack, since there is no need for an ordinary device to reroute this packet.

4.1.2 ARP Spoofing

ARP Spoofing, or ARP cache poisoning, is often used as a method to realize a MitM attack [37]. In an ARP Spoofing, the attacker tries to inject a malicious en-

try in the victim's ARP cache. The attack can be done in many different ways by faking ARP requests, ARP replies or both [57]. After a successful ARP Spoofing, the victim's ARP cache is corrupted and the MAC address of some other device is replaced with the MAC address of the attacker. This tricks the client into sending data to the attacker when the client is supposed to send data to some other device.

ARP Spoofing attacks could be hard to prevent, in particular when different operating systems handle ARP requests and replies differently. It could also be hard to know when an attack is made with certainty, since it is possible for IP addresses on the local network to change, making the ARP cache change as well. One possible way to discover an ARP Spoofing attack would be to launch a counter ARP Spoofing to detect if the suspected attacker is really rerouting the packages, such an attack is proposed by Z. Trabelsi and K. Shuaib [37]. This requires that the attacker itself is exposed to ARP Spoofing attacks. However, it is not considered good practice to launch an attack against another device.

The method used in this project uses more of a passive approach. By assuming that the ARP cache, in a normal use case, should be close to static, and only monitoring for changes in the ARP cache, it is possible to detect when the MAC address of a certain IP address is changed. Changes like this could indicate, at least if they are frequent, that a Man-in-the-Middle attack has been performed. This means that there is no need for sending any request to measure this entity, and there would be no need for implementing any extra protocols, which often is the case when handling ARP spoofing attacks [16] [36] [38]. The negative aspect of this method is that it could raise some false positives, but in the scheme of the entire situation-aware solution, this should be acceptable.

4.1.3 Promiscuous Mode

When a Network Interface Card (NIC) is set to Promiscuous Mode, it forwards all received information to the CPU rather than filtering out all information not meant for the device. For example in the Ethernet case, all packages with a MAC address not being the one of the device or any broadcast MAC address could be passed to the CPU. This means that a device with Promiscuous Mode activated can read data not intended for the device, i.e. sniff network data.

There exist ways to check whether a device in the local network has Promiscuous Mode activated, other than to locally access the device. Z. Trabelsi et al. and M. A. Qadeer et al. both propose a technique where ARP requests are used [48] [55]. In their proposal, a faked ARP request is sent to a target device to check if Promiscuous Mode is activated. The ARP request contains a fake MAC address as the recipient and the IP of the target. If Promiscuous Mode is not activated the package is rejected directly by the NIC. If instead Promiscuous Mode is activated the package is forwarded to the CPU. Depending on the OS of the target, the CPU often performs some kind of software filtering of the package. Although in most cases, destination addresses similar to the Ethernet broadcast address, such as FF:FF:FF:FF:FF:FE, are accepted [48]. If the package is accepted, a reply is

created and sent back to the sender. This means that if a response is received from this type of faked ARP request, it is highly likely that the device has Promiscuous Mode activated and is possibly performing a Sniffing attack.

4.1.4 Network Topology

When considering situation awareness, one of the most fundamental aspects is the network topology. The term does not only take the number of devices into account but also which type of devices and the connection between these devices [30]. Other things to include in network topology could be which services that are running on the network, DHCP servers or DNS servers just to name a few, and also how many neighboring networks the current network has.

Number of Devices

Scanning the number of devices on a local network can be done in many different ways, using the ARP ping method is one of the most efficient and reliable ways. This method broadcasts an ARP request for each possible IP in the network range and listens for a reply. Compared to the usual ICMP ping, it is a lot harder for a device to hide from an ARP ping since every host that wants to be reached by another host needs to have ARP enabled, except for the rare cases when static ARP tables are used. A problem with this method is that if the subnet range is large, the number of requests will also be large. IANA has specified three different ranges for private subnets, allowing for sizes of 65.356, 1.048.576 or 16.777.216 IP addresses [28].

Even the smallest number in this range, 65.356, would mean an equal amount of requests for scanning the entire network. This amount of requests would be very energy consuming and require a lot of processing power. Luckily, most subnets do not use these large ranges of possible IP addresses. A quick study of the three best-selling router brands, TP-Link, Netgear and Cisco, shows that their best-selling routers only have a subnet size of 256 IP addresses by default [27] [40] [50] [51]. Assuming that most routers designed for private use follow the same pattern, and that the default subnet size is not changed, makes the ARP Ping Scan a feasible method for scanning the network for devices.

Another method for calculating the number of devices on the network, is to simply listen and keep track of the number of ARP requests since they are broadcasted. From the ARP requests the sender's IP and MAC address can be stored, as well as the IP of the receiver. The advantage of this method is that there is no need for sending any extra requests, this saves power compared to ARP Ping Scan. The disadvantages are, however, that the device constantly has to listen for and keep track of incoming ARP broadcasts, which is costly in terms of processing power, and that the calculations may be inaccurate since not all hosts send ARP requests.

Network Services

When finding out which services a host is running, a port scan can be performed. One of the most used ways is a TCP Port Scan. This method sends a TCP package to the port, on the target device, that is to be examined with the SYN flag set. If an SYN-ACK reply is received, a service is known to be running on that port. If instead an RST-ACK is received, there is no service running. When it is known which ports that are in use, each port has to be mapped to a service. This can be done using a table with the most common services and their corresponding port number.

Scanning for ports can be a costly operation since there exist numerous well-known ports, with a corresponding service, to scan for each device [29]. With a typical subnet size of 256 IP addresses, the number of requests required for a complete scan of services can be very large. However, if only certain services are of interest, there is no need to scan all the ports available. For example, if only the number of SMTP and FTP services are of interest, only ports 25 and 21 have to be scanned, saving a lot of work and requests.

Another method of discovering running network services is to listen for incoming packages. This method is not valid for discovering all services since it requires some of the messages, to or from the service, to be broadcast so they can be sniffed by all devices. Example of protocols that can be sniffed is DHCP and IGMP. The advantage of this method is that it is more passive and does not require any extra requests to be sent.

4.1.5 Port Surveillance

Network ports are seen as a large attack surface for a device as it binds network data being sent, to an application. The recent Mirai botnet targeted certain ports when infecting a device with its malware [33]. Surveillance of ports can, therefore, give information in regards to the threat level of a device. Port surveillance should only be used as an indicator for threats, since there is a possibility that a device will receive incoming, non-malicious, requests for a port not in use. The ports chosen for this project include some which were targeted by the Mirai botnet and the Reaper botnet [49]. They can be seen in Table 4.1. Surveillance of ports assumes that no ordinary communication to the cameras is supposed to use these four ports, i.e. the device should not host any services on these ports.

Table 4.1: Well-known ports exposed to threats with corresponding Protocols.

Protocol	Port
SSH	22
Telnet	23
HTTP	80 and 8080
HTTPS	443

4.2 Bluetooth

This section presents situation parameters that are unique for Bluetooth. Even though Bluetooth is not directly related to IP, some IP Cameras still uses Bluetooth for setup and local access to the devices. It should, therefore, be taken into consideration when evaluating the threat situation for such devices [26] [35] [39].

4.2.1 Authentication Requests

The authentication process for a Bluetooth device is that of a challenge-response scheme. This scheme is meant to verify that a device is in the possession of a key specific to an earlier established link, called a Link key. While Bluetooth protects from brute-forcing the authentication, by waiting after a failed authentication before a new attempt may be done, it does not protect from sending repeating authentication requests. These requests will trigger a response from the other device which will be encrypted by the Link key. This could leak information regarding the key. Therefore, measuring the number of authentication requests could give an indication of a potential threat [43].

4.2.2 Network Topology

Since Bluetooth is an Ad-Hoc network the network topology can not be measured in the same way as a Wi-Fi network. Even so, the number of devices in the vicinity, and the services that these devices run, are still useful parameters.

Number of Devices

A Bluetooth device can be in three different modes of discovery: silent, non-discoverable, and discoverable. In silent mode the device only listens to traffic, never responding, and can therefore not be connected to. The two modes, non-discoverable and discoverable, both allows connections being made, where the difference is how "visible" the device is to other devices. If a device is in the discoverable mode, it will respond to device inquiries performed in its range, a non-discoverable device will not do this. The different discoverable modes makes it difficult to get an exact number of devices in the vicinity, when using the Bluetooth device inquiry. However, this is the only feasible method for this thesis, since other methods require great amounts of time or external hardware [18] [44].

Network Services

Bluetooth has a protocol specifically designed to get information of which services devices run. The Service Discovery Protocol (SDP), specifies which services are running on what device and how to connect to these services.

4.3 Shared Parameters

Even though there are differences between Wi-Fi and Bluetooth connections, some situation parameters are common for both connections. Such parameters could be

related to the sensitivity of the data rather than the transferred data. This section handles such parameters, and how they can be measured or interpreted, in order to evaluate the situation.

4.3.1 Data Sensitivity

An important consideration when determining which level of security the device should apply is the sensitivity of the data sent. For example, when using an IP camera, sensitive information could be the camera stream from a kitchen, living room or bedroom. Less sensitive data could come from a camera filming the garden, or other outside areas, since such information probably could be viewed by anyone.

The complication with data sensitivity lies in its measurement, where it is hard for a device to measure it on its own. The data sensitivity should preferably be defined by some pre-set rules and provided as input to this solution. Such rules for a camera could be:

- Is there anything moving in the picture - if there is, the data should be considered more sensitive.
- Is the camera filming an outdoor or indoor environment - the indoor environment should be considered more sensitive.
- Not all data coming from a camera is a video stream, it might also send metadata and device configuration which may have different sensitivity.

4.3.2 Packet Delays

Package delays can give useful information as to what a device is doing with the data. Measuring the delay between an incoming package, which is only supposed to be retransmitted, and an outgoing package can give an indication as to what the device does with the data. If the delay is long enough the monitored device could be altering the data maliciously. As this delay depends on many factors, such as the function of the device and the data load of the network, it should be used as an indicator only and no proof of malicious activity.

4.3.3 Integrity Protection

Integrity protection is used to protect sent data from changes during the transmission to the intended receiver, either unintentional or intentional. The integrity protection applied in this project provides protection against both intentional and unintentional changes. For the integrity protection to be useful, the receiving party must check the integrity protection of all received data.

It is not possible to determine whether the changes are intentional or unintentional. However, if they occur very frequently it indicates that there could be a malicious user altering the messages, or in some other way disturbs the communication. Either way, there is a reason for increasing the security level. Integrity protection is

a very good indicator to take into account since it requires no extra calculations or transmission and always has to be checked either way when receiving a message.

Chapter 5
Adaptation and Evaluation

The adaptive part of the solution connects the situation measurements to the different protection levels. The ability to adapt is what makes the solution useful, and it is of great value that it is correctly calibrated to match the measurements to different threat levels. Since the threats differ a lot with the use cases of each product, it should be up to the user or manufacturer of each product to decide how the situation parameters should be weighed in the decision making. Instead, the focus of this project lies in how often the situation measurements and adaptation should occur and what should be measured, as stated in Section 1.2.1. This paper will also propose two possible ways to weigh the situation parameters. However, this is only to be considered as two out of many possible ways to do so.

5.1 Mapping the Situation Parameters to Protection Levels

The proposals for this projects adaptive part splits the situation parameters into three different classes: Direct Threats, Indirect Threats, and Topology.

1. **Direct Threats** are the most serious threats and the countermeasures available are very efficient in preventing this threat. The idea is that the suspicion of one direct threat alone is enough to raise the protection level. An example of this type of threat is MitM attacks.

2. **Indirect Threats** are threats that are serious to the device but the threat is not directly related to the level of security of the sent data. Such threat could be a malicious user trying to access the situation-aware device instead of the data directly. In this case, a combination of topology and indirect threats or more than one indirect threat should be enough to raise the protection level.

3. **Topology**, which is the least serious class, is not related to any particular threat. Instead, it is used to give an idea of how the entire network is structured, and if there are any other devices on the network that could be exposed to risks. The topology alone is not enough to raise the security, but could be used together with direct threats or indirect threats to raise the protection level.

Table 5.1: Situation Classes and Threat levels for Wi-Fi.

Threat Classification	Situation parameter and threshold
Direct Threats	Promiscuous Mode more than 1 device (router)
	Data Sensitivity High
	ARP Cache Difference
	IP Routing more than 1 device (router)
	Package Delay over 15 ms
	Integrity Protection Failed
Indirect Threats	Attempted HTTP Connection
	Attempted HTTPS Connection
	Attempted Telnet Connection
	Attempted SSH Connection
Network Topology	IGMP Requests More than 1
	DHCP Requests More than 1
	ARP Request More than 1/sec
	Number of devices More than 5
	HTTP Service Present
	FTP Service Present
	SMTP Service Present

Two proposals are presented on how the situation parameters may be sorted, one for Wi-Fi and one for Bluetooth. These proposals can be seen in Table 5.1 and in Table 5.2

Table 5.2: Situation Classes and Threat levels for Bluetooth.

Threat Classification	Situation parameter and threshold
Direct Threats	Data Sensitivity High
	Package Delay over 15 ms
	Integrity Protection Failed
Indirect Threats	Authentication Requests More than 2/s
Network Topology	Number of devices More than 5
	HTTP Service Present
	FTP Service Present

5.2 Adapting Frequency

How often the situation is to be assessed and measured, is a key aspect when implementing a situation-aware scheme. The usefulness of such a solution is directly dependent on how often the situation is measured. The obvious trade-off here is how recent the situation measurements have to be, compared to the cost of performing the measurements. In order for the solution to be useful, the overall cost of the situation-aware part has to be small, compared to the cost of the protection. Naturally, this means that the situation measurements cannot be performed continuously, but instead has to be performed in intervals. The following different ways of implementing the measurement and adaptation part will be analyzed.

1. **Use of external device** - Use an external device, for example a sync-module, which does not run on batteries, to perform most of the heavy situation measurements and communicate the results to the cameras.
2. **Adapt before sending** - Since many cameras communicate very intensively during short time intervals where the video is sent, one possible adaptation scheme is to measure the situation prior to the device sending data.
3. **Fixed interval time** - One of the most simple approaches is to set a fixed interval time and at the end of each interval make new measurements and adapt to the situation.

Chapter 6
Throughput Tests

This chapter presents throughput tests for the algorithms introduced in Chapter 3. These tests are performed in order to get an early indicator on the comparative energy efficiency between the algorithms, where it is roughly assumed that a lower throughput indicates a more costly algorithm. These tests will also be the basis for a protection level proposal presented in Section 6.3. The tests performed in this chapter are performed only for the cryptographic functions alone, and not for the entire solution with a specific algorithm. This chapter introduces the importance of separation of libraries, this is followed by the throughput tests for the pure Python implementation as well as the C implementation, which used a Python wrapper.

6.1 Separation of Libraries

One of the advantages with Python is the large number of libraries available, including for example easy-to-use functions for encryption and decryption. Due to the small time frame of this thesis, and the fact that real-world applications often use functions from such libraries, it was decided that this project is to use as much of the already existing libraries as possible. This also decreases the risk of inaccurate results due to errors in the algorithms or inefficient implementations.

The Python interpreter allows for developers to write new modules for Python in C or C++. It is then possible to use these modules directly from Python [47]. This is in most cases a very useful feature which can be used for speeding up essential parts of a program. One of the most widely used cryptographic libraries for Python, pycrypto, uses this feature. However, this can cause a problem when comparing algorithms to each other, since the throughput of an implementation made in pure Python is usually slower than an implementation using C with a Python wrapper. An example of this is found in the results in Table 6.1. From here on, this thesis will refer to pure Python implementations as Python implementations. Likewise, a C implementation with a Python wrapper will be referred to as a C implementation.

Table 6.1 shows a comparison of a Python implementation and a C implementation. The Python implementation is about 2870 times slower than the C im-

Table 6.1: Python vs C Implementation.

Programming Language	Encryption Throughput [MB/s]
Python	0.002937
C	8.430

plementation. The C library used is well-known and heavily optimized for speed, which contributes to the big difference in throughput between the two. The extra time consumed by the Python wrapper for the C implementation is negligible in comparison to the time consumed by the actual encryption. It is worth noting that the Python implementations of the cryptographic algorithms in this chapter come from different libraries with different contributors. This can, of course, lead to some algorithms not being as optimized for speed as others. For this reason, it is only fair to keep Python implementations and C implementations separate when comparing speed and throughput. The results from the throughput tests in Python are to be considered as a guideline for how well the algorithms can perform, and what algorithms that are worth using for the different levels of security when implemented in C.

6.2 Throughput Test Result

To get an idea of how well different algorithms perform compared to each other, and as a first step towards implementing different levels of protection, throughput tests were performed. The different cryptographic algorithms are implemented in both Python and C. As stated in Section 6.1 these results are to be considered as a guideline for choosing suitable algorithms for the different levels of protection, rather than exact and accurate results for each individual algorithm. All tests ran on the same Raspberry Pi Zero W with an ARM11 1GHz processor and the Raspbian Stretch Lite operating system.

6.2.1 Encryption

A comparison of the encryption algorithms studied in Section 3.1, AES and SPECK, were performed to benchmark their throughput. The result of this test is shown in Table 6.2, the first one being the classic AES which is widely used in for example TLS, and the second being the lightweight-cryptographic algorithm SPECK, which is a lightweight algorithm developed for optimal speed in software.

Table 6.2 shows that the faster lightweight-cryptographic algorithm is more than four times as fast as the classic AES256 cryptographic algorithm when using the Python implementation. The difference when using the C implementation is not as large as with the Python implementation or in the literature study [21]. However, the throughput for SPECK128 is still more than double the throughput for AES256. The results clearly show that there is a possible advantage by using the faster SPECK in a situation were throughput is of great essence.

Table 6.2: Encryption throughput tests.

Algorithm	Encryption Throughput in Python [kB/s]	Encryption Throughput in C [MB/s]
AES256	2.937	8.430
SPECK128	13.463	19.252

6.2.2 Hash Algorithms

Since hashes will be used for integrity protection, some of the most common hash algorithms were also compared. The result is shown in Table 6.3. All hash functions used the same output size of 256 bits.

As expected, the lightweight-hashes BLAKE2b and BLAKE2s performed better than the widely used SHA3 algorithm in both Python and C. BLAKE2s were almost three times as fast as SHA3-256 in C, and twice as fast in Python. What was more unexpected was that SHA2-256 was faster than BLAKE2b, and almost as fast as BLAKE2s, in the C implementation. This is not the case in the Python implementation, where BLAKE2b is the fastest algorithm, over three times as fast as SHA2-256. This means that the relationship between two algorithms may vary depending on which language they are implemented in.

Table 6.3: Hash throughput tests.

Algorithm	Hash Throughput in Python [kB/s]	Hash Throughput in C [MB/s]
SHA3-256	2.491	5.1193
SHA2-256	4.650	14.237
BLAKE2b	15.957	12.825
BLAKE2s	8.504	15.477

6.2.3 Message Authentication Code

Also connected to integrity protection is a MAC. This is for cases when no encryption is used, but integrity protection is still needed. The tests compared three algorithms, the classic HMAC using the SHA2-256 hash function, as well as the keyed versions of the two BLAKE variants, BLAKE2b and BLAKE2s. The HMAC-SHA2-256 and the two BLAKE algorithms all produce a 256-bit output.

Just as with the hash functions, HMAC-SHA2-256 seems to perform significantly better in C than in Python, compared to the other algorithms. Another similarity with the hash functions is that BLAKE2s performs better in C than BLAKE2b but worse in Python. Despite this, the differences between the three algorithms are

quite small, with only a 20% difference in throughput between the best algorithm, BLAKE2s and the worst algorithm, BLAKE2b, in C.

Table 6.4: Python MAC throughput tests.

Algorithm	MAC Throughput in Python [kB/s]	MAC Throughput in C [MB/s]
HMAC-SHA2-256	4.592	13.259
BLAKE2b keyed	15.385	11.833
BLAKE2s keyed	8.959	14.178

6.2.4 Comparing encryption, hash, and message authentication

The algorithms considered most secure for each area, encryption, hash, and message authentication, were as expected also the ones with the lowest throughputs in Python. The results were similar for C, with BLAKE2b being the only exception. Interestingly, the most costly encryption algorithm, AES256, had roughly the same throughput as the most costly hash algorithm, SHA3-256. Besides the fact that the C implementations are over 1000 times faster, there are a few important differences between the C implementation and the Python implementations in the previous chapter. The difference in throughput, between AES and SPECK, is smaller with SPECK128 being only twice as fast as AES. Furthermore, BLAKE2s is the fastest hash and message authentication algorithm instead of BLAKE2b, and the C implementation of SHA2-256 performed much better compared to BLAKE2 than the Python implementation.

These results indicate that, although the algorithms perform more similarly in C than in Python, it is possible to create different protection levels with a distinct difference in throughput, and presumably power consumption as well, for both C and Python implementations.

6.3 Protection Levels Proposal

Following the throughput results of the different cryptographic algorithms, and the evaluation of these results, protection levels were created. The levels used for this project along with cryptographic algorithms for each level are seen in Table 6.5.

Table 6.5: Proposed Protection Levels.

Protection level	Cryptographic Algorithm	Protection Achieved
HIGH	AES256 with SHA3-256	Confidentiality and Integrity
MEDIUM	SPECK128 with BLAKE2s	Confidentiality and Integrity
LOW	BLAKE2s keyed	Integrity
NONE	None	Clear text

Table 6.6: Throughput for the proposed Protection levels.

Protection level	Throughput in Python [kB/s]	Throughput in C [MB/s]
HIGH	1.348	3.185
MEDIUM	5.212	8.580
LOW	8.959	14.178

Table 6.6 shows the throughput for the cryptographic algorithms used by the protection levels in Table 6.5. It is clear that the levels have a distinct separation in throughput for both implementations. Also, note that this evaluation takes both the hash calculation and encryption time into account. The time it would take to encrypt the extra hash for the two highest protection levels is not considered. However, this extra time will be insignificant, at least when sending long messages, in comparison to the total encryption time since the extra data to encrypt is only 256 bits. The protection level NONE has been omitted from Table 6.6 since this protection level does not use any cryptographic algorithms.

Chapter 7

Power Consumption Tests for Protection Levels

To compare the different protection levels in a more realistic use case, power consumption tests were carried out on a Raspberry Pi Zero W running Raspbian Stretch Lite. To measure the power consumption, the Raspberry Pi was connected to a USB multimeter, which is able to measure consumed energy.

7.1 Python Implementation

During the tests, the Raspberry Pi was encrypting, hashing, and sending data at a fixed bit rate of 1.44 kB/s, this rate ensures that the processor usage is close to 100% when using protection level HIGH. Because of the very low bitrate, the cost of sending was insignificant. This means the results for Bluetooth and Wi-Fi were almost identical and they are therefore presented together as one result. The results of these tests are found in Table 7.1. As a reference running the Raspberry Pi in idle mode without sending any data at all requires 622 mW.

The column named normalized power consumption is the power consumed for protection and sending, minus the power consumed in idle mode. This is equal to the power consumed by the Raspberry Pi for encrypting, hashing, and sending the data alone. Since most IP Cameras have a much lower power consumption when being idle it is most fair to compare the normalized power consumption, instead of the average power consumption, in order to determine how much power that can be saved. Blink states for example that their IP camera is able to last for up to 5 years with only two AA batteries [11]. With the best kind of AA batteries the authors could find, about 3500 mAh per battery, this would mean an idle state with a power consumption of less than 1 mW [23] [24].

There are a number of ways to reduce the power consumption of the Raspberry Pi such as disabling the LED or the HDMI port etc. Some reports state that these changes could reduce the power to about 80 mW [25]. However, since this is not considered to be in the scope of this project, such things have not been done to optimize the power consumption in idle mode.

Table 7.1: Python Encryption and Send Tests.

Protection level	Processor Usage	Average Power Consumption [mW]	Normalized Power Consumption [mW]	Normalized Energy Consumption per kB [mJ/kB]
HIGH	96%	846	224	156
MEDIUM	30%	694	72	50
LOW	15%	658	36	25
NONE	<1%	622	<1	<1

7.2 C Implementation

The same tests were then repeated with the same protection levels implemented in C. The biggest difference from these tests compared to the Python tests is that the sending now stands for a significant part of the processor usage and the consumed power of the device. The results differed up to 10% when performing the same test twice, making the results inaccurate. When repeating the tests without the sending part, the different individual measurements were the same for the same tests. The sending part could differ a lot from one measurement to another. In order to achieve more accurate results, the cryptographic part was carried out without sending any data. Later, a longer test was carried out to get an average cost of sending the data. This makes the different tests more reliable since now the average cost for sending data can be added to get an estimated cost of data sending and protection.

The results from the encryption tests are shown in Table 7.2 below. To increase the processor usage the data rate was increased to 2.88 MB/s, over 2000 times more than the Python case. Note that the results in this table do not include the sending of data.

Table 7.2: C implementation Encryption Tests.

Protection level	Processor Usage	Average Power Consumption [mW]	Normalized Power Consumption [mW]	Normalized Energy Consumption per MB [mJ/MB]
HIGH	90%	947	325	113
MEDIUM	60%	798	176	61
LOW	20%	694	72	25
NONE	<1%	622	<1	<1

7.2.1 Sending

As previously mentioned, longer tests were carried out in order to get an average cost of sending data. This was done for both Wi-Fi and Bluetooth with a data rate of 2.88MB/s and 20kB/s respectively. The energy consumption for the different sending methods can be seen in Table 7.3. Table 7.4 shows the cost of encrypting

Table 7.3: Sending data, Wi-Fi and Bluetooth.

Network	Energy Consumption per MB [mJ/MB]
Wi-Fi	101
Bluetooth	875

and sending, for both Wi-Fi and Bluetooth, where the result from Table 7.3 was added to the results in Table 7.2. Note that the values here are not actually measured, but instead an addition of two measured values in order to achieve more reliable results.

Table 7.4: Energy Consumption for Encrypting and Sending data.

	Energy Consumption per MB [mJ/MB]	
Protection Levels	Wi-Fi	Bluetooth
HIGH	214	988
MEDIUM	162	936
LOW	126	900
NONE	101	875

7.3 Power Consumption Evaluation

The results of the Python tests in Table 7.1 shows that there is a 28% decrease in energy consumed when using no protection at all compared to using the highest level of encryption. This value is equal to the highest possible gain using a situation-aware solution with this kind of idle power. In absolute values, this corresponds to 238 mW. Comparing the normalized energy consumptions, the results are even better. This is because of the costly Python implementation where the cost of sending is negligible. The overall results for the protection levels also look very promising with distinct differences in power consumption between the levels. Despite this, the C implementation is preferred over the Python implementation. With higher throughput and lower energy consumption, further implementations will be focused on the C implementation.

The result for the C implementation looks similar to the Python results. The maximum saved energy for a situation-aware adaptive solution is 18% with the

idle power consumption of the Raspberry Pi. The reason this number is lower than the Python implementation is that C is more efficient, which results in lower costs for encrypting the same amount of data. The sending of data also counts for a significant part of the energy, with 101 mJ/MB for Wi-Fi and 875 mJ/MB for Bluetooth. Sending and encrypting data with Bluetooth on this device is clearly worse than with Wi-Fi. Where the maximum saved power for Wi-Fi is 53% whereas the maximum saved power for Bluetooth is just 11.5%. The main reason for the disappointing Bluetooth result is the low send rate. The Raspberry Pi W used, maximized the data rate at 20kB/s, when using the library pybluez, which is low.

Chapter 8

Situation Measurements

This chapter presents the results of two different tests on the framework which measures the situation parameters. The first test measures the time it takes for each measurement to run. The second test is a power consumption test, similar to those in Chapter 7. These two tests was performed for both Wi-Fi and Bluetooth connections.

8.1 Wi-Fi Measurements

This section handles all tests performed on Wi-Fi networks, meaning parameters from Section 4.1 and Section 4.3. The results from these measurements are later compared to the results from Section 8.2 in Section 8.3

8.1.1 Test setup

The network simulated when testing the situation parameters can be seen in Figure 8.1. This network is created to look like a typical home network. In the figure, the IP camera represents the device running the situation-aware framework. It is connected to a network where there is another device sending data and running services such as SMTP, HTTP, and FTP to simulate a real network of ten devices, which is considered to be a reasonable number for a home network. The IP Camera is simulated by a Raspberry Pi Zero W with an ARM11 1GHz processor and running the Raspbian Stretch Lite operating system.

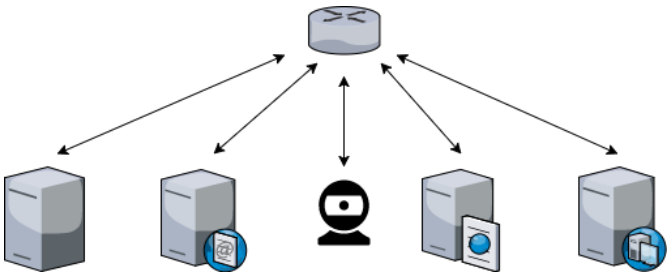

Figure 8.1: Test Network Setup.

8.1.2 Passive Measurements

Assessing the threat level of a network requires measurements from said network. The measurements can be divided into two parts, a passive one and an active one. The passive part contains broadcasts from different servers or incoming requests which the device continuously monitors. Table 8.1 describes which requests that are monitored through the passive measurements. In this case, the first three protocols are related to network topology and the communication is broadcasted to all devices in the network. The last four protocols are related to more direct communication with the devices. If the measurements show any activity of the last four protocols it indicates that someone has tried to communicate or access this particular device.

Table 8.1: Passive Situation Measurements.

Protocol	Identified by	Address of Receiver
ARP	EtherType 0x0806	Multicast
IGMP	Internet Protocol Number 2	Multicast
DHCP	Port 68	Multicast
SSH	Port 22	Unicast
Telnet	Port 23	Unicast
HTTP	Port 80 and 8080	Unicast
HTTPS	Port 443	Unicast

8.1.3 Time Consumption for Active Measurements

The second part is the active part which means that the device has to send requests or execute code locally on the machine in order to get measurements. The device listens for replies from the sent requests and then evaluates the result. This part is not feasible to run continuously since it requires data to be sent, which costs a lot of energy. The different implemented active measurements are found in Table 8.2.

Table 8.2 shows the number of bytes sent during the test and the time for each test. The table also shows how the number of bytes sent will scale if the number of devices or the subnet size would change. It is reasonable to assume that the time for the measurements would scale in a similar manner to the number of bytes sent.

8.1.4 Power consumption for Situation Measurements

The power consumption tests for the situation measurements were done in a similar manner to the protection levels in Chapter 7 on the same low-powered device. These measurements are shown in Table 8.3. The purpose of these measurements is to get an idea of how costly each measurement is, except for the passive measurement which runs continuously. All measurements, except for the passive and the ARP Ping Scan, ran for one hour with ten measurements every second. The ARP

Table 8.2: Active Situation Measurements

Measurement	Bytes sent	Scaling factor for number of bytes	Time [ms]
ARP Cache Differences	0	None	1.7
IP Packet Routing Activated	640	Linear - ND	39
Promiscuous Mode Activated	640	Linear - ND	62
Package Delay	128	None	5.0
ARP Ping Scan	16320	Linear - SS	440
SMTP Service Scan	640	Linear - ND	38
HTTP Service Scan	1280	Linear - ND	76
FTP Service Scan	640	Linear - ND	38

ND stands for Number of Devices.

SS stands for Subnet Size.

Ping Scan ran 5 times every other second, since it requires a timeout to decide whether a device is present for every possible IP address in the subnet. Average power consumption is the power consumption for the entire test and normalized is the average power consumption minus the power consumption in the idle state. Energy Consumption per measurement is the mean consumption for one measurement.

The last measurement in this table is all the other measurements combined together. This can be seen as a complete situation-aware measurement of all parameters. These complete measurements ran only once every second instead of every ten times per second due to the slow nature of the Raspberry Pi. The normalized energy consumption per tests is, however, still the mean energy consumption for running all the combined measurements once.

Table 8.3: Situation Measurements Tests for Wi-Fi.

Situation Measurement	Test Frequency	Average Power Consumption [mW]	Normalized Power Consumption [mW]	Normalized Energy Consumption per measurement [mJ/meas.]
Passive	continuously	625	<1	<1
ARP Cache Difference	10/s	622	<1	<1
IP Packet Routing	10/s	679	57	5.7
Promiscuous Mode	10/s	684	62	6.2
Package Delay	10/s	642	20	2.0
ARP Ping Scan	2.5/s	684	62	24.8
SMTP Service Scan	10/s	645	23	2.3
HTTP Service Scan	10/s	668	46	4.6
FTP Service Scan	10/s	645	23	2.3
All combined	1/s	658	36	36.0

8.2 Bluetooth Measurements

This section covers the measurements for the situation parameters for Bluetooth from Section 4.2 and Section 4.3.

8.2.1 Test Setup

The test setup for the Bluetooth case is similar to the Wi-Fi, with an identical Raspberry Pi Zero W running the Raspbian Stretch Lite OS. The exception, in this case, is the lack of a Wi-Fi connection. Instead, the devices ran all the tests over Bluetooth. Since Bluetooth does not follow the same structure as Wi-Fi, it is hard to control the number of active Bluetooth devices in proximity of the device. The number of nearby devices stretched from about 1 to 10 devices during the tests, which seems to be a reasonable level for a typical home environment.

8.2.2 Bluetooth Measurements

The Bluetooth measurements are quite different from the ones performed on a Wi-Fi network. Bluetooth is an Ad-Hoc network where the devices pairs as master/slave without any central node. Performing measurements like Device Scan requires an inquiry to be sent over an extended time which increases the power consumption of such a measurement. Table 8.4 shows the time it takes to perform each active measurement. It is worth noting that these measurements are independent of the number of Bluetooth devices in proximity to the device performing the test.

For the energy consumption test, the parameters have been divided into passive and active measurements as can be seen in Table 8.5. The inefficiency of the

Table 8.4: Active Situation Measurements.

Measurement	Time
Device Scan	11.5s
Service Scan	12s
Package Delay	0.3s

active measurements is due to the time it takes to perform a complete Device and Service Scan, having to scan for up to 12 seconds to get reliable results, shown in Table 8.4. While the idle power consumption is low when running Bluetooth compared to Wi-Fi, at 510 mW compared to 622 mW, the active measurements are still not very energy efficient. The passive measurements are far better with a power consumption being almost indistinguishable from when the device is idle. The results of the measurements can be seen in Table 8.6.

Table 8.5: Parameters divided into Passive and Active measurements.

Parameter	Type
Data Integrity	Passive
Data Sensitivity	Passive
Authentication Requests	Passive
Device Scan	Active
Service Scan	Active
Package Delay	Active

Table 8.6: Situation Measurements Tests for Bluetooth.

Measurement	Test Frequency per hour	Average Power Consumption [mW]	Normalized Power Consumption [mW]	Normalized Energy Consumption per measurement [mJ/meas.]
Passive	continuously	525	15.5	N/A
Device Scan	352	536	25.8	260
Service Scan	187	536	25.8	496
Package Delay	7200	695	185	93

8.3 Situation Measurements Evaluation

In order to create a successful situation-aware solution, it is important to evaluate the situation parameters. Since measuring the situation may require a lot of energy,

it is important to only use the most relevant parameters to receive a high possible gain when using situation-aware scheme instead of a classic cryptographic system, such as TLS.

8.3.1 Time Consumption for Wi-Fi Evaluation

When looking at the time and the number of bytes sent from Table 8.2 it is clear that the ARP Ping Scan is a heavy operation, where the number of bytes sent exceeds all the other measurements combined. It is therefore preferred to use the ARP Ping Scan as little as possible, for example only at startup and later use the ARP cache, the number of ARP requests, or the DHCP broadcasts, as a way to estimate the number of devices on the network. This is also somewhat a "best case scenario" for the ARP Ping Scan since the subnet size was only 256 IP addresses large during the tests. In a worse scenario, the subnet size could be 65.536 addresses, which means that over 4 MB of data need to be sent to perform a single ARP Ping Scan.

The most promising result from the Table 8.2 is the ARP Cache Differences. Using this method requires 0 bytes to be sent and this is also by far the fastest option, which may be because there is no need for waiting for any replies from other devices. The table also shows that the Delay Time measurement is a good option. This measurement does not scale in either time or data sent and is independent of network size and number of devices in the network. For all other active measurements, the number of bytes sent is linear to the number of devices in the network. This means that for networks with few devices connected, it could be reasonable to use all of the measurements more often. In networks with a lot of devices, prioritizing when it comes to what measurements to use as well as when and how often to use them, must be done.

8.3.2 Power Consumption for Wi-Fi Evaluation

The first two measurements from the Table 8.3, Passive and ARP Table Comparison, are the only two tests which do not require any data to be sent at all. This is, of course, an advantage since extra energy is consumed when sending data. The power consumption for a single ARP Table Comparison was so small that it was not measurable. The negative aspect of the passive measurement, compared to all other measurements, is that it has to run continuously or at least over a longer period of time in order to be useful. This may be costly, in particular on devices with very low idle power. Another important note regarding the passive measurements is that the difference between running the passive measurements, and not running anything at all, is so small that it was difficult to measure. For that reason, it is hard to draw any exact conclusions regarding the passive measurements other than that the extra energy consumption is small in this particular case.

It is surprising that the SMTP and FTP Service Scans are more efficient than the IP Packet Routing and the Promiscuous Mode measurements. Since they require the same amount of data to be sent, they should be similar in energy consumption.

The most likely reason for this is due to implementation differences in the tests. While IP Packet Routing and Promiscuous Mode measurements used a raw socket and manually crafted packages, the Service Scans used a prewritten stream socket which presumably is more efficient [34].

Adjusting the frequency of the scans is a way of lowering their overall power consumption and it should be done in consideration to the energy consumption per measurement. The most expensive measurement is by far the ARP Ping Scan, which consumed twice as much energy as the HTTP Service Scan, and is the first measurement which frequency should be considered.

Running all the tests combined is less expensive than running all the individual tests, about 75% of the cost. This is due to the Raspberry Pi consuming less energy when sending all the requests closely together compared to sending the requests one at a time.

In Chapter 5 the situation parameters researched in this thesis were divided into three categories in order of their relevance to the different threats. The highest order, named Direct Threat, is comprised of Promiscuous Mode, Data Sensitivity, ARP Cache Difference, IP Routing, Package Delay and Integrity Protection. While Data Sensitivity and Integrity Protection do not consume additional power, since it is a provided input, the other parameters do. The cost for the ARP Cache Difference was so low that it was not possible to get any accurate results with the instruments used, making it a great candidate. The measurement for Promiscuous Mode, ARP Cache Difference, IP Routing, and Package Delay each consume under 7 mJ as seen in Table 8.3. While still quite costly these measurements should still be highly prioritized since they are related to direct threats.

The second highest order, Indirect Threats, consists of the passive measurements HTTP, HTTPS, Telnet, and SSH. Although the power consumption is low for these measurements, they need to be done somewhat continuously, as mentioned earlier, which could lower the effectiveness of the solution.

The last classification is that of Network Topology. As detailed in Section 5.1, the measurements are made to get a sense of the overall risk of the network. These parameters include IGMP, DCHP, and ARP of the passive measurement as well as the ARP Ping Scan and the SMTP, HTTP, and FTP Service Scan. Although the passive measurement does not consume much power, the ARP Ping Scan and the Service Scans do. The ARP Ping Scan alone accounts for more energy than all of the other measurements combined. Seeing as this network topology classification does not necessarily give an indication of a threat, it is by far the least efficient group of situation measurements.

If trimming of the situation measurements are needed in order to become more energy efficient, the classification Network Topology is a prime candidate. However, many of the other active measurements need a list of the devices on the network,

Table 8.7: Classification of Situation Parameters.

	Direct Threat	**Indirect Threat**	**Topology**
Passive	Integrity Failed Data Sensitivity	HTTP Connection HTTPS Connection Telnet Connection SSH Connection	ARP Requests DHCP Requests IGMP Requests
Active	Promiscuous Mode IP Routing Package Delay ARP Cache Diff.		Device Count HTTP Service FTP Service SMTP Service

which ARP Ping Scan provides. One solution to this problem could be to remove the ARP Ping Scan completely and only use the listener and the ARP table to create a list of the devices on the network. The disadvantage with this solution is of course that this list would not necessarily include all devices on the network. Instead of removing the measurements completely an adjustment of the frequency could also be considered.

One of the most important things when creating a situation-aware solution, as well as one of the questions to research from Section 1.2.1, is to evaluate which situation parameters to measure. This is important since the cost of the measurements greatly impacts the usefulness of the entire solution. A summary of all situation parameters is found in Table 8.7. Naturally, it is desired to use the most relevant and cheap situation parameters. This table can be seen as a general guideline for which order to implement the different measurements starting from the class Direct Threats - Passive Measurement. The last class to use should be the Network Topology - Active Measurement class. In between this, there is a trade-off between relevance and measurement cost when deciding which parameters to use.

8.3.3 Bluetooth

In Section 8.2 the energy consumption of the situation measurements for Bluetooth are shown in Table 8.6. Comparing these results with the ones from the Wi-Fi tests, presented in Table 8.3, it is clear to see that the Bluetooth measurements are very costly. Running all of the Bluetooth situation measurements is 23 times more expensive than running all the Wi-Fi situation measurements. As mentioned before, this is due to the low Bluetooth send rate for the Raspberry Pias well as the amount of time the measurements have to run. The results of the Bluetooth tests, both sending and situation measurements, are disappointing and point to the fact that Bluetooth is not suitable for the purpose of this thesis. Even though the idle power of only running Bluetooth is significantly lower than running Wi-Fi, the cost of sending and measuring the situation is too high. Note that this result is device specific and a situation-aware Bluetooth solution may very well be suitable for other applications.

Chapter 9

Protection and Situation-awareness Comparison

This chapter connects the results from Chapter 7 and 8, and evaluates the entire solution as a whole. Three criteria to be met in order to successfully implement a situation-aware adaptive cryptography scheme are stated. Following this is an evaluation and discussion of the gathered result against these criteria. Lastly, a full system test of the most suitable implementation is presented.

9.1 Situation-aware Adaptive Cryptography Criteria

To get an idea of how feasible a situation-aware approach is, it is important to compare the cost of the situation-aware part with the cost of encrypting and sending data. This will determine whether it is useful to implement adaptive handling of security protocols or not, which is one of the purposes of this thesis, as stated in Section 1.2.1. If the solution is to add any real value to a product, it must either save processing power or energy, preferably both, compared to a standard encryption solution such as AES256 with a SHA3 hash. Other results would mean a less secure system at a higher cost. In order to succeed with a situation-aware adaptive cryptography solution, three basic requirements have to be met.

1. The sensitivity of the sent data must be of a level where it is acceptable to not use the highest level of protection at all times. In other words, there has to be a possible trade-off between security and cost in terms of throughput and energy consumption.

2. The different protection levels, or security protocols, must differ in terms of cost. This cost must be significant in terms of the total cost of the device being idle and sending data combined.

3. The total cost for the measuring, adapting to the new situation, and changing protection level must be small in comparison with the cost that can be saved by using another protection level. This cost is directly related to the second question to research from Section 1.2.1.

9.2 Data Sensitivity

The first requirement depends on every individual use case and should be up to the manufacturer or user of each device to consider, before implementing or using a situation-aware adaptive scheme. In the case of IP Cameras, there is no doubt that situations exist where data always should be protected with the highest level of security available. On the other hand, for private use of such cameras, there exist a lot of situations where the protection level can be lowered for an improvement in battery time and throughput.

9.3 Difference Between Protection Levels

The results from Chapter 7 shows that it is possible to meet the second requirement when using Wi-Fi. From comparing Table 7.2 and Table 7.3, it is seen that the cost for encrypting 1 MB data with the highest protection level is roughly as costly as sending 1 MB of data. For the protection level LOW, the cost of sending is about four times as high as the cost of protection. Even though these test results are only valid for the specific platform they ran on, it is highly likely that the cost of protection is significant for a variety of low powered IoT devices and not least IP Cameras.

Bluetooth proved to be, on this device, too costly when measuring the situation and sending data, compared to encryption. This can be seen in Table 7.4 where the sending of data, at the protection level HIGH, accounted for 88.5% of the total energy consumption for encrypting and sending. Failing this requirement makes Bluetooth unsuitable for the situation-aware adaptive cryptography scheme studied in this thesis, and will not be considered further in this report. Furthermore, no system tests for Bluetooth was carried out.

9.4 Cost of measuring

To determine whether the third criterion is met, it is important to first determine which device that is to perform the situation measurements and how often the measurements are performed. The most straightforward way is to let each IP Camera perform all measurements individually. In that case, the total cost of measuring, adapting, and changing protection level would be directly dependent on how often the measurements and adaptation were carried out. Another case is to use a master-slave relationship, where a number of cameras are connected to each other via for example Wi-Fi. In this case, it would not be necessary for all cameras to perform the same tests. Instead, only one device could perform the tests, and then communicate the results of the tests to the rest of the group. This would probably increase the power that could be saved, but it would also increase the complexity of the solution. The cameras would, for example, have to decide which camera that has to run the tests. Another approach for connecting a number of cameras is with the use of an external device, such as a Sync Module, similar to what Blink uses [12].

9.4.1 External Device

The advantage of using an external devices, such as a Sync Module, is that such a module does not need to be battery powered. With a non-battery powered sync module, most of the situation measurements could be performed by the sync module instead, without having to take the cost of the measurements into consideration. The only measurements that would have to be done by the cameras would be the passive measurements, where the camera listens for incoming unicast requests. This could be done by the sync module as well, but some threats may then remain undetected. Since the power consumption for the passive measurements is very low, the total power consumption of the situation measurements would be almost negligible. The result of this would be that only the first and the second requirement, for useful situation-aware systems, need to be taken into account. However, the timing of the measurements must be considered. If the measurements were to be performed before every communication with the cameras, the time to perform the tests would cause a bottleneck. A better solution would be to perform the measurements continuously, with a rather short time interval, since the power consumed by the tests is of less importance.

9.4.2 Adapting before sending

Looking at the case where every camera performs the entire set of situation measurements individually, it differs quite a lot from the Sync module case. The optimal solution of measuring and adapting will differ with the different use cases. Taking the Blink camera as an example, where Blink has defined standard usage as "approximately 10, 5-second video event per day" where every 5-second clip requires up to 750 kB of data [11] [13]. According to Table 7.4, protecting and sending 750 kB of data over Wi-Fi would require about 161 mJ with the highest protection level, and about 76 mJ without any protection, making the maximum possible gain 85 mJ per 5-second clip. If a complete measurement, with all the parameters, were to be done, the cost of that would be about 36 mJ. Exactly how much energy would be saved, if any, depends on the calculated protection level. Table 9.1 shows how much energy that would be saved depending on the protection level.

Table 9.1: Changes in energy when adapting before sending.

Protection Level	Energy per Sent Clip with Situation-Aware solution [mJ]	Change in percent from default (161 mJ)
HIGH	197	+22.4%
MEDIUM	158	−1.9%
LOW	131	−18.6%
NONE	112	−30.4%

The results look quite promising for the implementation of a situation-aware solution, since energy is saved in three out of four cases. If a camera would do a complete situation measurement before every sent clip, which must be considered as very good conditions for the camera to assess the situation, the worst case scenario would require 22.4% extra energy. The best case scenario, on the other hand, would save 30.4% energy. In between this, the proposed solution would save either 1.9%, or 18.6%, depending on the determined protection level. To improve the results, ARP Ping Scan could be removed from the situation measurements as proposed in Section 8.3. This would reduce the total energy cost to about half of the previous cost for the measurements, changing the worst case scenario to 11.2% more energy consumption, and the best case scenario to a reduced power consumption of 41.6%.

The change in percent from Table 9.1 is directly dependent on the amount of data sent after each situation measurement. With smaller amounts of data sent, the possible savings will decrease, and the risk of a higher energy consumption will increase. With larger amounts of data were to be sent, the possible overhead will turn to zero, making a situation-aware solution a great option. Figure 9.1 shows the different worst-case scenarios depending on the data size sent, where the ARP Ping Scan is not included.

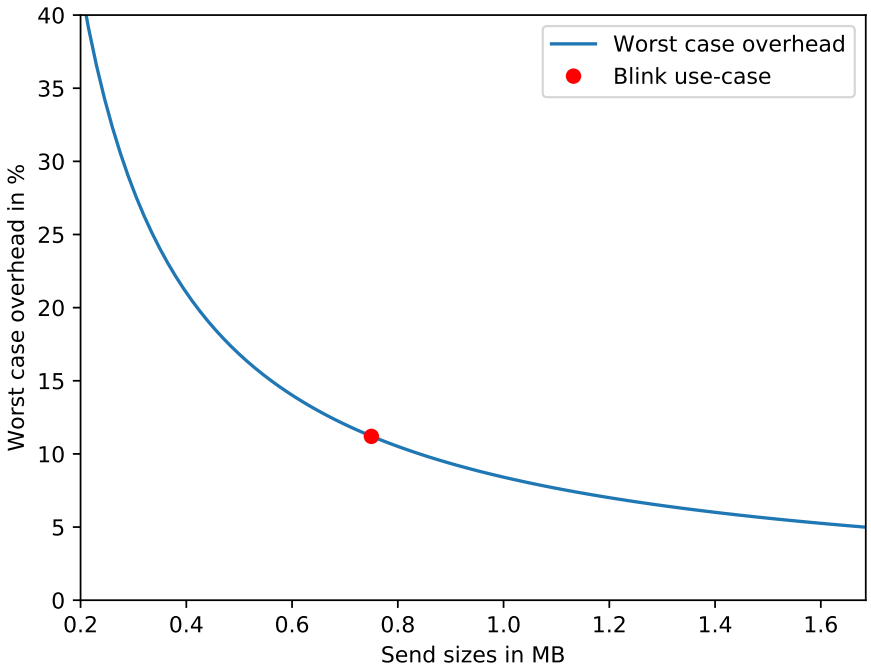

Figure 9.1: Maximum overhead depending on data sent.

The absolute limit of this situation-aware solution is a send size of 160 kB, counting without the ARP Ping Scan. With send sizes below this number, the possible gain is below zero. It is possible to change this limit by omitting even more measurements, such as scanning for services before every sending, which is discussed in Section 8.3. But even without the Service Scans, the absolute limit of a situation-aware solution would be a send size of about 80 kB. For IP Cameras, which is the device considered for this thesis, 80 kB is a small amount of data since a single picture can be a few MB. In other cases, where lower data rates are used, it would not be so obvious that a situation-aware solution is a feasible choice, even if the ARP Ping Scan was not used. Example of such cases could be a smartwatch, a smart smoke detector, or some other device where the typical data sizes are lower than those for an IP Camera.

A disadvantage of using the approach where measurements are done right before the camera sends data, is the response time of the camera. If, for example, a user requests data from the camera, the user must wait for a situation evaluation to take place before the camera can reply. A complete situation evaluation for the device takes about 700 milliseconds, which should not be considered critical in most use cases. If instead a situation measurement without ARP Ping Scan and Service Scan would be performed, the connection time would be reduced from 700 milliseconds to about 108 milliseconds which should be low enough to not cause a bottleneck in most applications. Another solution to this issue is to perform the situation evaluation with some time interval in order to always have an idea of the situation.

9.4.3 Adaptation with interval

The idea of interval adaptation is to evaluate the situation regularly, in order to always be ready to communicate without the delay that occurs when the device is measuring the situation before every sending of data. Another advantage is that the device can send multiple small messages without having to adapt to the new situation, which may not be particularly useful for an IP Camera, but perhaps for a smoke detector, a sensor or some other device that only uses smaller amounts of data to communicate. A disadvantage of this method is that the latest measurements may not be up to date with the current situation when the device sends the data, since the situation may vary quickly.

This method of adaptation can be realized in many different ways. The most intuitive way is to set a fixed time interval for the situation measurements, for example once every hour. Looking at the Blink use case again, which states that standard use is “approximately 10, 5-seconds video events per day” where each video clip is up to 750 kB. The changes in energy with these assumptions are found in Table 9.2. The results are worse than those in Table 9.1, and in all four use cases the total energy is increased, making the situation-aware solution a bad choice for this use case. These results are of course directly dependent on how often the situation is measured, and how much data that is transferred. If instead the costly ARP Ping Scan was omitted from the measurements, the worst case scenario would be

Table 9.2: Changes in energy when adapting with fixed interval.

Protection Level	Average Energy per Sent Clip with Situation-Aware solution [mJ]	Change in percent from default (161 mJ)
HIGH	247	+53.7%
MEDIUM	208	+29.1%
LOW	179	+11.2%
NONE	161	+0.4%

improved to a 26.8% overhead and the best case scenario to a 26.1% reduction in energy, making the implementation somewhat useful.

Changing the interval time for which the situation is measured also changes the possible gains with a situation-aware solution. A longer time period yields better possible gains, but the risk that the measured situation would not be valid at the time of the data transfer, would increase. The limit for how often the measurements can be done, if ARP Ping Scan is omitted, is about 2 times per hour. If the measurements were to be done more frequently, this would result in no possible gain at all.

Another way of implementing interval adaptation, would be to let the time to the next measurement depend on the results of the last measurement. If, for example, the situation has changed a lot since the last time, or if the threat level is high, it could be of greater value to evaluate the situation more often than if the situation were unchanged at a very low level. A way to implement this is to double the interval time between measurements, up to some limit, if the protection level is unchanged between two measurements. If the protection level was changed, reset the time between measurements to its default value, for example 1 hour. This would result in a potentially lower total cost for measurements but a not so up-to-date situation assessment if the situation would change suddenly.

The ways to realize interval adaptation are many. The methods presented here are just some of the most straightforward ways to do so. More complex ways, using artificial intelligence or machine learning, could possibly provide even better results. In the end, it is still a trade-off between how up-to-date the situation measurements are required to be and their cost. The same goes for the different ways of implementing a situation-aware solution, using either a sync module, adaptation before sending, or interval adaptation, where the optimal solution is up to every use case. A sync module may be the best solution if the extra cost for a new module is not a problem and the module can be supplied with power. Adapting before every sending of data could be the best choice if the response time of the unit is not critical. Interval adapting may be the best choice if the energy consumed by the measurements is not a critical part of the total energy consumed

by the device.

9.5 System Tests

In order to get an idea of the efficiency of the solution in a typical use case, system tests were carried out for Wi-Fi. A closer look at the implementation of the complete solution is found in Appendix B. Two tests were done, one to simulate a typical home environment without potential threats, and one to simulate an environment where a threat was present. From these tests, it is possible to see how much energy that is saved, and how well the suggested adaptation proposal from Section 5.1 performs. The first test used a capture file, with real network data from the Network Solutions company AppNeta, to simulate a home environment [4]. The threats were simulated using the Python library Scapy, which offers a wide range of custom crafted packages [53]. The results from this system tests are shown in Table 9.3.

Table 9.3: System Tests.

Environment	Average Measured Threat Level	Change in energy consumption from standard solution (AES-256 SHA-3)
Threats present	3.03	−1.2%
No Threats present	1.37	−26.3%

The second column, "Average Measured Threat Level", ranges from 1 to 4 and is meant to reflect the different protection levels. The number 1 corresponds to the protection level NONE and 4 correspond to using the protection level HIGH when sending data. The system tests use the Blink typical use case from Section 9.4.2 and adapts every time before sending data.

Both cases from Table 9.3 reduce the power consumption which is a good result. In the case of no threats present, the average protection level is between NONE and LOW and the power reduction is about 26%. This seems reasonable for a typical home environment. In the case of a threat present, the protection level is at the level MEDIUM most of the time. Even though this results in a power reduction, the presence of a threat should raise the threat level higher. This is still not an immediate security flaw since MEDIUM uses secure cryptographic functions to encrypt and integrity protect the data which are all unbroken. However, when a threat is present it is, of course, suggested to use the highest level of protection available. This indicates that the protection proposal from Section 5.1 may need calibration in order to better respond to threats.

Chapter 10
Conclusions

This thesis has investigated and implemented a complete situation-aware solution, which has the ability to reduce the power consumption for energy constrained devices, in a successful way. After measuring the proposed protection levels, it was found that the implemented solution could save up to 52.8% of the power used for sending and encrypting data. In a typical use case, stated by Blink [11], the solution could save up to 30.4% even if the complete set of situation parameters were used and the situation was measured every time before sending of data. These results could be even better if the implementation of some situation parameters would be more optimized. For example, the Promiscuous Mode and the IP Routing measurements are about 3 times as expensive as the SMTP or FTP Service Scans while they all require the same number of bits to be sent. This indicates that there is room for improvement for the Promiscuous Mode and IP Routing implementations. Such improvements would, of course, improve the overall results of the solution.

Looking at the aim and questions to research for this thesis from Section 1.2 and Section 1.2.1, the purpose of this thesis is fulfilled. A set of situation parameters for determining a threat level and what to react to has been proposed in Chapter 4 and the evaluation of these parameters is presented in Chapter 8. How often the situation can be measured and adapted to, as well as the circumstances under which a situation-aware solution is possible, has been investigated throughout the thesis and are discussed in Chapter 9.

10.1 Future work

This section contains the authors suggestions for future work. This could be used to deepen the knowledge in the area of situation-aware adaptive cryptography and further learn where, and when, it can be used.

10.1.1 Hardware Implementation

This thesis has investigated a number of different cryptographic algorithms and compared their performance in throughput and power consumption. All of these

algorithms have been implemented in software only. A lot of devices today implement cryptographic algorithms in hardware for improvements in speed, and therefore also energy consumption. It would be very interesting to see if the results from this thesis would still hold for implementations in hardware. The results would probably not be as promising, since the total cost for encrypting the data would be reduced, making the relative cost of the situation measurement bigger. Another aspect with hardware implementations is of course that one device would have to implement multiple cryptographic schemes, which probably would increase the production cost and maybe even the size of such a device. The solution to this problem could be to use one single cryptographic function for all the protection levels and simply adjust the number of rounds for the different levels of protection.

10.1.2 Real Network Data

Another interesting aspect would be to try the solution out and calibrate it using real network data from typical home networks containing cameras. From these results, it would be possible to calculate the average saved power consumption, perhaps in a more accurate way than the one presented in Section 9.5. Since the threat level for most home networks probably is low most of the time, the level of protection could also be low most of the time. Access to real network data would also make it possible to compare a situation-aware solution with doubling time interval between the situation measurements, which is described in Section 9.4.3, or machine learning approaches to other methods such as adapt-before-send or adapt with a fixed interval time more accurately. Comparing the methods to each other, without access to real network data, would not produce accurate results.

10.1.3 Key Establishment

One thing that has been left out of this thesis is the key exchanges and establishments. While this thesis assumes pre-distributed keys, this is often not the case in real world applications. The fact that situation-aware adaptive cryptography requires quite a lot of changes between different cryptographic functions makes the cost of the key exchange particularly relevant. If the key was to be changed every time a new protection level is chosen, the effectiveness of the situation-aware solution would be decreased. Another approach is to use the same master key for all the different cryptographic functions and use for example different hashes of that key to achieve the appropriate key lengths for the cryptographic functions. A comparison of different key exchange methods, including lightweight versions, was published by R. Alvares et al. [3] and the results seem promising for using in a situation-aware solution, such as the one proposed in this thesis.

10.1.4 Bluetooth

The threats concerning Bluetooth communication are many and due to the Ad-Hoc structure they can be a lot harder to detect. This thesis has just scratched the surface of a wide range of Bluetooth vulnerabilities. It would have been interesting to try out some of the dedicated hardware, such as the Bluefruit LE Sniffer, for Bluetooth sniffing in order to improve the device discovery part [2]. Even though

the size of the Bluefruit sniffer is very small, adding external hardware always increases the power consumption.

It is also of interest to investigate Bluetooth further on platforms other than the Raspberry Pi Zero W which has been used for this project. The very low send rates made the situation-aware solution infeasible for this particular platform. The bitrates measured from the Raspberry Pi using the Pybluez library is far from the the maximum for the particular Bluetooth version used [45]. Implementing the solution on a platform with higher bitrates for Bluetooth could, possibly, make the situation-aware scheme a feasible solution.

References

[1] F. Abed, E. List, S. Lucks, J. Wenzel, (2013), *Cryptanalysis of the SPECK Family of Block Ciphers*, Cryptology ePrint Archive, Report 2013/568, https://eprint.iacr.org/2013/568

[2] Adafruit, (2017), *Bluefruit LE Sniffer - Bluetooth Low Energy (BLE 4.0) - nRF51822 - v3.0* https://www.adafruit.com/product/2269 (Accessed 2018-02-02)

[3] R. Alvarez, C. Caballero-Gil, J. Santonjaand A. Zamora, (2017), *Algorithms for Lightweight Key exchange*, Sensors (14248220),DOI: 10.3390/s17071517

[4] AppNeta, (2018), *Sample Captures*, http://tcpreplay.appneta.com/wiki/captures.html (Accessed 2018-02-01)

[5] J. P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, C. Winnerlein, (2013), *BLAKE2: simpler, smaller, fast as MD5*, Cryptology ePrint Archive, Report 2013/322, https://eprint.iacr.org/2013/322

[6] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, (2013), *Simon and Speck Families of Lightweight Block Ciphers*, Cryptology ePrint Archive, Report 2013/404, https://eprint.iacr.org/2013/404

[7] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, (2013), *Simon and Speck Families of Lightweight Block Ciphers*, page 2, Cryptology ePrint Archive, Report 2013/404, https://eprint.iacr.org/2013/404

[8] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, (2015), *SIMON and SPECK: Block Ciphers for the Internet of Things*, Cryptology ePrint Archive, Report 2015/585, https://eprint.iacr.org/2015/585

[9] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, L. Wingers, (2015), *The SIMON and SPECK lightweight block ciphers*, Design Automation Conference 52nd ACM/EDAC/IEEE, ISBN: 9781479980529

[10] Blink, (2016), *Is the system secure? What security measures do you have in place?*, https://blinkinfo.desk.com/customer/en/portal/articles/2299237-is-the-system-secure-what-security-measures-do-you-have-in-place- (Accessed 2018-01-15)

[11] Blink, (2017), *CBlink Technical Specifications*, http://support.blinkforhome.com/customer/en/portal/articles/2292825-blink-technical-specifications (Accessed 2018-01-04)

[12] Blink, (2018), *Blink Sync Module*, https://blinkforhome.com/products/blink-sync-module (Accessed 2018-01-11)

[13] Blink, (2018), *Blink Frequently Asked Questions*, https://blinkforhome.com/pages/frequently-asked-questions (Accessed 2018-01-11)

[14] A. Bogdanov, D. Khovratovich, C. Rechberger, (2011), *Biclique Cryptanalysis of the Full AES*, Advances in Cryptology – ASIACRYPT, DOI: 10.1007/978-3-642-25385-0_19

[15] A. Bogdanov, (11 August 2011), *AES Encryption Cracked*, https://www.theinquirer.net/inquirer/news/2102435/aes-encryption-cracked (Accessed 2017-11-07)

[16] D. Bruschi, A. Ornaghi, E. Rosti, (2003), *S-Arp A secure Address Resolution Protocol*, Computers Society Applications Conference Proceedings 19th Annual IEEE, DOI: 10.1109/CSAC.2003.1254311

[17] CNSS, (2015), *Use of public standards for the secure sharing of information among national security systems*, CNSS Advisory Memo 02-15

[18] D. Cross, J. Hoeckle, M. Lavine, J. Rubin, K. Snow, (2007) *Detecting Non-Discoverable Bluetooth Devices*, IFIPAICT volume 253, ISBN: 9780387754611

[19] CRYPTREC, (2013), *Specifications of e-Government Recommended Ciphers*, Japan Ministry of Internal Affairs and Communication, http://www.cryptrec.go.jp/english/method.html (Accessed 2018-02-28)

[20] Department of Defence, (2001), *Department of Defense Dictionary of Military and Associated Terms* https://web.archive.org/web/20091108082044/http://www.dtic.mil/doctrine/jel/new_pubs/jp1_02.pdf (Accessed 2018-02-27)

[21] W. Diehl, F. Farahmand, P. Yalla, J. P. Kaps, K. Gaj, (2017), *Comparison of Hardware and Software Implementations of Selected Lightweight Block Ciphers*, 27th International Conference on Field Programmable Logic and Applications, DOI: 10.23919/FPL.2017.8056808

[22] T. Dierks, E. Rescorla, (2008), *The Transport Layer Security (TLS) Protocol - Version 1.2*, https://tools.ietf.org/html/rfc5246 (Accessed 2018-01-15)

[23] Energizer, (2018), *ENERGIZER L91*, http://data.energizer.com/pdfs/l91.pdf (Accessed 2018-01-04)

[24] Energizer, (2018), *ENERGIZER E91*, http://data.energizer.com/pdfs/e91.pdf (Accessed 2018-01-04)

[25] J. Geerling, (2015), *Raspberry Pi Zero - Conserve power and reduce draw to 80mA*, https://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-zero-conserve-energy (Accessed 2018-01-09)

[26] W. Greenwald, (2018), *The Best Home Security Cameras of 2018*, http://uk.pcmag.com/webcams-products/39333/guide/the-best-home-security-cameras-of-2018 (Accessed 2018-01-26)

[27] M. Harans, (2016), *The Top 9 Best-Selling Router Brands In Q4 2016*, http://www.crn.com/slide-shows/networking/300083524/the-top-9-best-selling-router-brands-in-q4-2016.htm (Accessed 2017-11-22)

[28] IANA, (1996), *Address Allocation for Private Internets*, https://tools.ietf.org/html/rfc1918 (Accessed 2018-01-15)

[29] IANA, (2011), *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*, https://tools.ietf.org/html/rfc6335 (Accessed 2018-01-15)

[30] D. Ince, (2013), *A Dictionary of The Internet, 3rd edition*, Oxford University Press, ISBN: 019280460X

[31] INMCM, (2018), *Implementations of the Simon and Speck Block Ciphers* https://github.com/inmcm/Simon_Speck_Ciphers, (Accessed 2018-02-01)

[32] J. A. Jerkins, (2017), *Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code*, IEEE 7th Annual Computing and Communication Workshop and Conference, DOI: 10.1109/CCWC.2017.7868464

[33] C. Kolias, G. Kambourakis, A. Stavrau, J. Voas, (2017), *DDoS in the IoT: Mirai and other Botnets*, Computer 50(7):80-84 2017 IEEE, DOI: 10.1109/MC.2017.201

[34] Linux man-pages Project, (2017), *Socket Documentation*, http://man7.org/linux/man-pages/man2/socket.2.html (Accessed 2018-01-25)

[35] Logitech, (2018), *Logitech V-R0008 - Specifications*, http://support.logitech.com/en_us/product/circle-2-home-security-camera/specs (Accessed 2018-01-26)

[36] W. Lootah, W. Enck, P. Mc Daniel, (2007), *Tarp: Ticket based address resolution protocol*, Computer networks, vol. 51, no. 15, DOI: 10.1109/CSAC.2005.55

[37] J.S. Meghana, T. Subashri, K.R. Vimal, (2017), *A survey on ARP cache poisoning and techniques for detection and mitigation*, Fourth International Conference on Signal Processing, Communication and Networking, DOI: 10.1109/ICSCN.2017.8085417

[38] S.Y. Nam, D. Kim, J. Kim, (2010), *Enhanced arp: Preventing arp poisoning-based man-in-the-middle attacks*, Communications Letters IEEE, vol. 14, no. 2, DOI: 10.1109/LCOMM.2010.02.092108

[39] NEST, (2018), *Nest Cam Outdoor - Tech Specs*, https://nest.com/cameras/nest-cam-outdoor/tech-specs/ (Accessed 2018-01-26)

[40] Netgear, (2015), *AC1750 Smart WiFi Router*, http://www.downloads.netgear.com/files/GDC/R6400/R6400_UM_07Aug2015.pdf (Accessed 2017-11-22)

[41] NIST (2001), *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*, Federal Information Processing Standard Publication 197

[42] NIST, (2015), *NIST Releases SHA-3 Cryptographic Hash Standard*, https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard (Accessed 2018-01-04)

[43] NIST, (2017), *Guide to Bluetooth Security*, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf (Accessed 2018-01-22)

[44] M. Ossmann, D. Spill, M. Ryan, W. Code, J. Boone, (2018), *Ubertooth*, https://github.com/greatscottgadgets/ubertooth/ (Accessed 2018-02-02)

[45] I. Paul, (2010), *Wi-Fi Direct vs. Bluetooth 4.0: A Battle for Supremacy* https://www.pcworld.com/article/208778/Wi_Fi_Direct_vs_Bluetooth_4_0_A_Battle_for_Supremacy.html (Accessed 2018-02-02)

[46] Pycryptodome, (2018), *Pycryptodome Documentation* https://pypi.python.org/pypi/pycryptodome (Accessed 2018-02-01)

[47] Python Software Foundation, (2017), *Python 2.7 documentation - Extending and Embedding the Python Interpreter*, https://docs.python.org/2.7/extending/index.html#extending-index, (Accessed 2017-11-14)

[48] M.A. Qadeer, A. Iqbal, M. Zahid, (2010), *Network Traffic Analysis and Intrusion Detection Using Packet Sniffer*, Second International Conference on Communication Software and Networks, DOI: 10.1109/ICCSN.2010.104

[49] Radware, (2017), *Reaper Botnet*, https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/reaper-botnet/ (Accessed 2017-11-28)

[50] Routers-Reset, (2017), *TL-MR3020*, https://routers-reset.info/2309/reset/TP-Link/TL-MR3020 (Accessed 2017-11-22)

[51] Routers-Reset, (2017), *Meraki - Z1*, https://www.router-reset.com/reset-manuals/Meraki/Z1 (Accessed 2017-11-22)

[52] R. Sandip, H. Tamzidul, B. Abhishek, B. Swarup, (2016), *The power play: Security-energy trade-offs in the IoT regime*, IEEE 34th International Conference on Computer Design (ICCD), DOI: 10.1109/ICCD.2016.7753360

[53] Scapy, (2018), *Scapy Documentation* https://scapy.readthedocs.io/en/latest/ (Accessed 2018-02-01)

[54] SecurityGem, (2017), *Best Home Security Camera*, https://www.securitygem.com/best-home-security-cameras/ (Accessed 2018-01-15)

[55] Z. Trabelsi, H. Rahmani, K. Kaouech, M. Frikha, (2004), *malicious sniffing systems detection platform*, International Symposium on Applications and Internet, DOI: 10.1109/SAINT.2004.1266117

[56] Z. Trabelsi, K. Shuaib, (2006), *Man in the Middle Intrusion Detection*, IEEE Conference on Global Communications, DOI: 10.1109/GLOCOM.2006.282

[57] M. V. Tripunitara, P. Dutta, (1999), *A Middleware Approach to Asynchronous and Backward Compatible Detection and Prevention of ARP Cache Poisoning*, Computer Security Applications Conference (ACSA99) Proceedings 15th Annual Phoenix UK, DOI: 10.1109/CSAC.1999.816040

[58] Wikipedia, (2018), *Man in the middle attack*, `https://upload.wikimedia.org/wikipedia/commons/thumb/e/e7/Man_in_the_middle_attack.svg/260px-Man_in_the_middle_attack.svg.png`, (Accessed 2018-02-08)

[59] Wikipedia, (2018), *S-Box* `https://upload.wikimedia.org/wikipedia/commons/c/c5/S-Box.png` (Accessed 2018-02-08)

[60] Wikipedia, (2018), *Substitution Permutation Network* `https://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/SubstitutionPermutationNetwork2.png/360px-SubstitutionPermutationNetwork2.png` (Accessed 2018-02-08)

Appendix A

Cryptographic Components

To get a deeper knowledge of how block ciphers are constructed, and what parts of a block cipher that are desirable to not use in lightweight ciphers, two components used in most block ciphers are explained in this section.

S-box

An S-box (Substitution Box) is a common part of many block ciphers and is used for introducing non-linearity. It produces an m-bit output from an n-bit input using a lookup table. The table can either be fixed or dynamically generated from, for example, a key. Figure A.1 shows an S-box where 4 inputs are matched to 4 output values. The implementation of S-boxes can be quite complex and

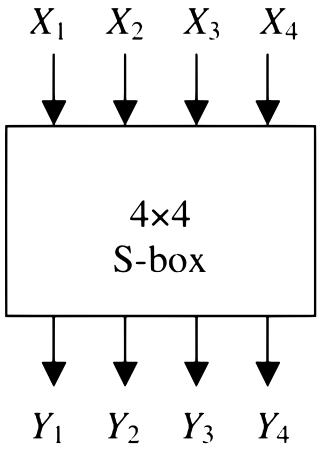

Figure A.1: S-Box [59].

costly, but they are desired since they allow for relatively easy security arguments. For more lightweight solutions it is preferred to instead increase the one-time work of doing cryptoanalysis, instead of the work done by devices with high constraints on encryption and decryption [8].

Substitution-Permutation Network

Substitution-permutation network (SPN) is a scheme of mathematical functions, typically used in block ciphers such as AES. SPNs are often used for encrypting plaintext using a key together with alternating rounds of substitution and permutations. Figure A.2 below shows an SPN with 3 rounds of 16 bits.

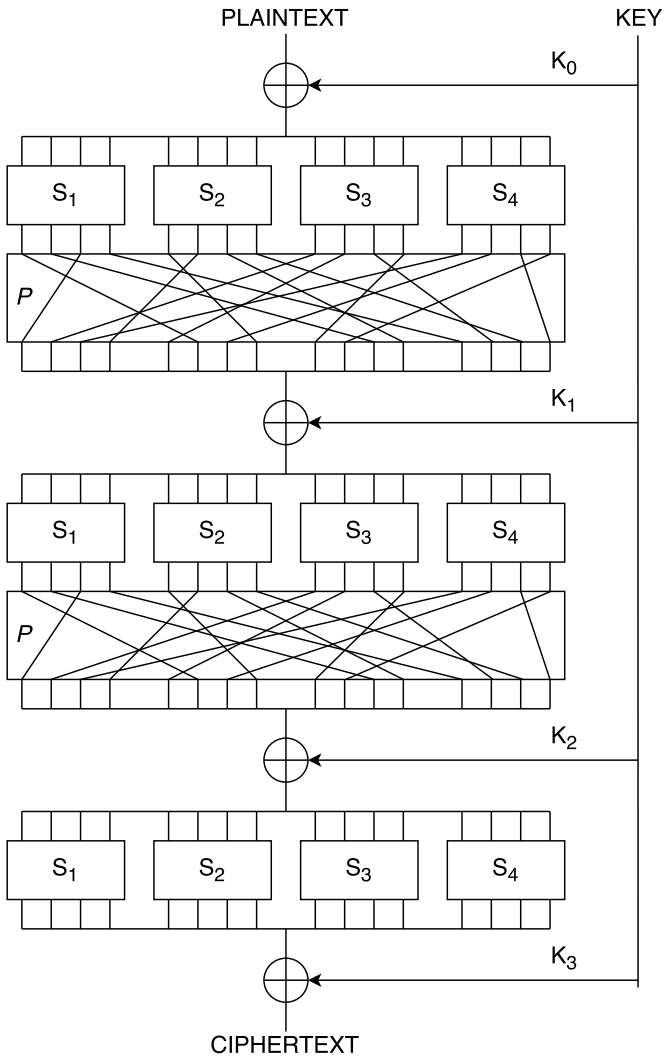

Figure A.2: SPN Network [60].

Appendix B

Overview of Complete Implementation

As stated in Section 6.1, this project has used as much of the already existing libraries in Python as possible. Figure B.1 shows an overview of the complete implemented solution, together with the used libraries and the programming language they are implemented in. This solution was used for the system test in Section 9.5. The use of wrappers in Python allows for writing the time-critical parts of the code in C and keeping the noncritical parts in Python for easier use. The socket interface has been used for the sending of data, and for the situation measurements which require requests to be sent. The protection of the data uses mainly the PycryptoDome library [46]. The exception is the level MEDIUM which also uses a SPECK implementation done by INMCM, with some modifications, since Pycryptodome lacks this [31]. The arrows, linking the different libraries together into a complete solution, are also written in Python.

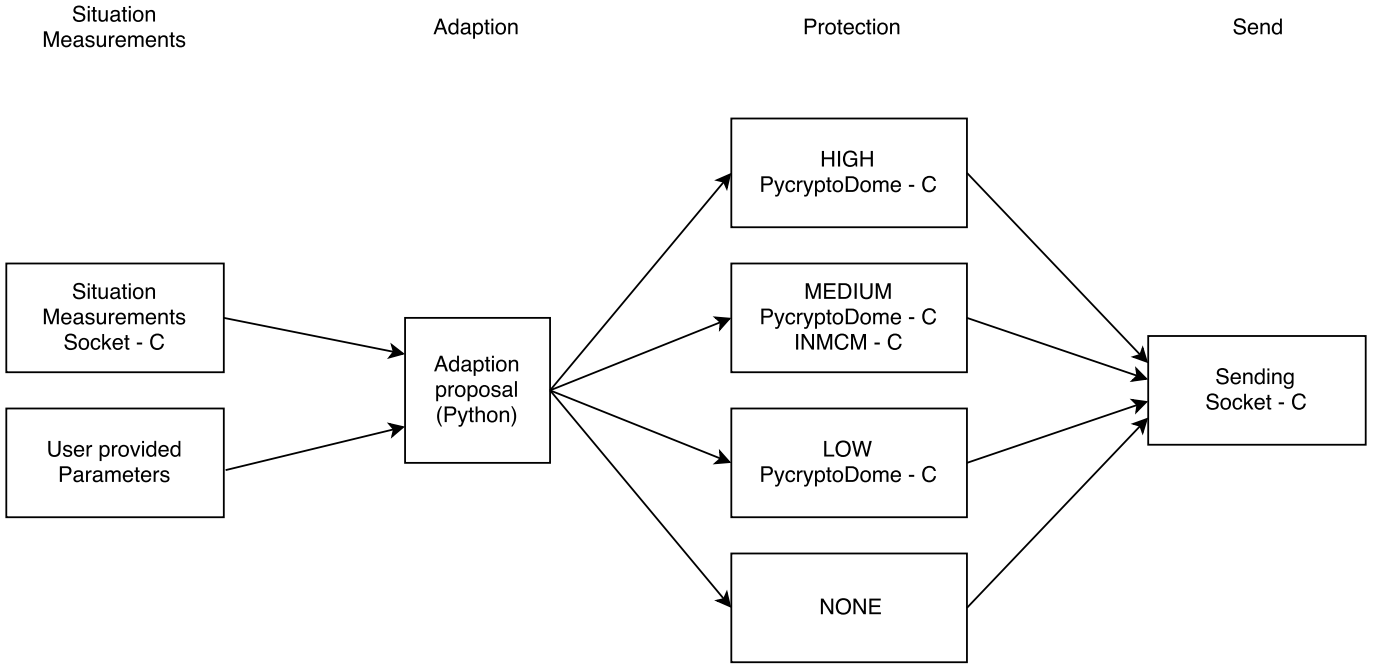

Figure B.1: Overview of the architecture for the complete implementation.

Printed by Tryckeriet i E-huset, Lund 2018

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2018-619

http://www.eit.lth.se