University of Twente

Faculty of Electrical Engineering, Mathematics and Computer Science

Master thesis

Security for an Internet-of-Things Network in the Context of a Dairy Management System

---

Author
D.T.R. Ikink

Committee
dr. A. Peter
dr.ing. F.W. Hahn
dr.ing. Y. Huang

Supervisors
dr.ing. F.W. Hahn
dr.ir. I. Wassink

December 17, 2020

UNIVERSITY OF TWENTE.

Abstract

Securing data in an Internet-of-Things network can be a challenge. Devices often operate on a very limited energy budget and thus common cryptographic solutions can quickly become too resource-hungry. This study provides a cryptographic system design focused on use in an Internet-of-Things network. The system design has been developed in the context of a precision dairy farming product, and runs on an EFR32MG22 microcontroller. Throughout this research we focus on identifying the performance gain that hardware acceleration of certain operations offers. We first perform an analysis of potential adversaries and the specific threats they pose. This was done using a combination of the STRIDE and DREAD frameworks for threat identification and ranking, as well as with a novel attacker profiling model that integrates with these frameworks. Next, we evaluate the performance of different symmetric encryption ciphers to determine which provides the most security whilst consuming as little energy as possible. We find that hardware-accelerated `AES-CCM` with key length of 256 bits consumes only roughly 4.5 mAh over a time span of ten years. To finalize our system design, we implement an identity-based key agreement protocol and evaluate its performance. This implementation shows that such a protocol is very suitable for this type of network, regardless of whether hardware acceleration is used or not.

Contents

Acknowledgements

I would like to express my sincere gratitude to my supervisors Florian Hahn and Ingo Wassink for their limitless insights, and guidance. Florian and Ingo shared my enthusiasm for this research and contributed to its completion in many ways.

I also wish to thank Jacqueline Schuurmans and Jeroen van Dijk for providing me with all the knowledge and tools I required for this research.

Finally, I wish to acknowledge all the support I have received from my girlfriend, friends, and family. Doing this research from home was rough at times, but you all were there to keep me motivated.

1 Introduction

The Internet-of-Things, often abbreviated as IoT, is the largest network of globally connected devices. Many cars, watches, and even refrigerators nowadays are either directly or indirectly (via an internet gateway) interconnected in a global network of "things". According to IoT Analytics, the total number of connected devices was over 9.5 billion in 2019 [1]. This number is expected to reach 28 billion by 2025. With the increasing size of the IoT network comes an increasing amount of data volume. Statista [2] forecasts that by 2025, an annual total of 79 zettabytes[1] will move through the Internet-of-Things. For comparison, in 2018 this was "only" 13.6 zettabytes. Much of this data comes from devices that are very deeply integrated into people's personal lives. Moreover, seemingly simple devices can often reveal a lot of information about who you are and what you are doing. Consider, for instance, smart energy meters. As McDaniel and McLaughlin describe in their work, smart meters have the potential unintended consequence of leaking valuable information about their users' habits and behavior [3]. Turning on a television or microwave creates a spike in energy consumption, signaling that you are home. Conversely, very low energy readings during the day might indicate that you are not home. It is clear that the IoT provides unprecedented opportunities for both legitimate and illegitimate operations. Unfortunately, the security of IoT devices is often subpar, in large due to the constrained nature of the devices [4]. Many of the conventional encryption techniques are based on mathematical principles that require more processing power and energy than IoT devices can offer. This remains one of the largest challenges in IoT security.

This research has been carried out in the context of a dairy management system (DMS). This system provides an all-in-one software and hardware package for farmers to monitor their dairy livestock. Every animal is equipped with a small device that contains a set of different sensors. The data that these sensors gather is transmitted to a local processing device where it is analyzed. Users of the DMS operate a medium to large-sized farm with one hundred to two thousand dairy cows. Keeping track of all these individual animals is a time-consuming task. Additionally, manual heat[2] detection is often less accurate as well. The cost of missing cow heat can be as high as $3 per cow per day [5]. That might not seem like much, but in a farm with thousands of animals the costs can quickly add up. Keeping the animals in good condition is a top priority for a farm business.

A deployed DMS network can grow large and a lot of potentially valuable data passes through it. Currently, this data is not properly secured. It is evident that, given the ever-increasing number of cyber attacks [6], securing your data is as important as having a functioning system. It is not difficult for a passive attacker to eavesdrop on what is happening inside the network. An active attacker could even go as far as injecting new data, potentially damaging a farmer's business and livelihood. Considering the constraints imposed by the computing power of the devices in the network, not all types of solutions work equally well. The most computationally restricted devices in a DMS network, the Animal Trackers, lack the resources to perform complex and time-consuming computations.

[1] 1 ZB = 1,000,000,000,000 GB

[2] The short period of time where a female animal is sexually receptive

As such, an effective yet practical solution is required. A second motivation for this research is to gain more control over who can and cannot access the data that flows through a DMS network.

The goal of this research is to design and implement a cryptographic system to secure data in a network of constrained devices. The main requirement for this system is that it uses as little energy as possible, while providing sufficient security. The details of such a system, as well as the process of designing it, are described in this report. We first provide an overview of a DMS network and the components it consists of in Section 2. Our actual research has been split up into four distinct parts. In Section 3 we identify and analyze potential adversaries and threats that this DMS might have to deal with in the future. We use a combination of established methods and our own novel addition to model and describe adversaries. In Section 4 we evaluate the performance of eight encryption schemes using different combinations of parameters. We first describe how we measured the performance of every cipher and then present our findings. We identified which combination of encryption scheme, mode of operation, and key size provides the most security whilst consuming as little energy as possible. Finally, in Section 5 we implement and evaluate a key agreement protocol using identity-based elliptic curve cryptography. Although it is a fairly new branch of cryptography, identity-based cryptography has many benefits over more traditional methods, such as a public-key infrastructure. In this final section we establish whether it is practically possible to use such a scheme in a constrained IoT environment.

2 System Overview

2.1 Definitions

- **DMS**: Dairy Management System
- **PU**: Processing Unit
- **UHF**: Ultra-High Frequency

2.1.1 System Components

A DMS network consists of two different components. The following section breaks down the roles and distinct features of the two components.

2.1.2 Animal Tracker

Cows wear a tag device called an Animal Tracker. Each Animal Tracker contains a set of sensors. The data from these sensors is used by other components of the system to perform a wide array of tasks, such as animal identification & location, and tracking their behavior.

2.1.3 Processing Unit

The central component in the network is the Processing Unit, or PU. All the data that is gathered by the Animals Trackers is transmitted wirelessly to the PU. The PU then analyzes this data and provides a convenient interface for the user to monitor and manage his livestock.

2.2 Network Topology

From a network perspective, a DMS network is a large, centralized network of small, low-power devices. It can potentially contain hundreds of small devices and uses multiple communication types and protocols. Evidently, a DMS network is really similar to what is known as an Internet-of-Things network. Although not all individual devices have a direct internet connection, they all use a gateway to communicate with an internet-connected backend. This IoT likeness means that a lot of related research in that field can be applied here. The Animal Trackers use radio communication with a frequency in the Ultra-High Frequency (UHF) range to broadcast sensor data. These data broadcasts are picked up by PUs. There is no set limit on the number of Animal Trackers that a single DMS network can contain. However, at some point, the UHF band becomes so cluttered with messages that it becomes virtually impossible to extract actual data due to interference.

2.3 Constraints

There are a number of clear constraints that our proposed solution must take into account, the most glaring of which is inherent to it being an Internet-of-Things: battery life. An

Animal Tracker has nominal capacity of only 2,000 mAh. For comparison, the average smartphone in 2018 was already able to store approximately 3,500 mAh in a single charge [7]: that is one and a half times the capacity that an Animal Tracker can work with. Moreover, a smartphone can be charged daily, whereas an Animal Tracker is expected to run for approximately ten years on a single charge. Clearly, there is very little room for computational redundancy and our solution must take note of this.

Another consequence of the low battery capacity is the fact that an Animal Tracker's UHF radio receiver cannot draw too much power. Research has shown that increased receiver sensitivity is related to increased power consumption [8]. This means that an Animal Tracker cannot have a high receiver sensitivity. The communication from a PU back to an Animal Tracker therefore becomes less reliable and thus weaker as distance increases. Data flowing from a PU to an Animal Tracker can only reach half as far as data coming from an Animal Tracker. This constraint imposes a few challenges for the proposed solution, because communication from a PU to an Animal Tracker should be reduced to a minimum. A handshake might be necessary to set up the communication between devices, because devices need some form of shared knowledge. In a symmetric encryption setting, this is the key that the devices use to encrypt their data. Another option is the use of pre-shared knowledge between the devices. The drawback to this approach is the fact that if the pre-shared knowledge is ever compromised, all past communication is compromised.

3 Threat Assessment

The first section of this research considers threat assessment. More specifically, it discusses the inclusion of an attacker profile into a well-known threat modeling framework. There exists a plethora of risk and threat modeling frameworks. Xiong et al. show this in [9] by assessing 176 research articles focused on risk modeling. They found that the landscape of state-of-the-art threat modeling frameworks is diverse and lacks common ground. Some models use a qualitative approach, and others a quantitative. A number of articles propose a graphical solution, whereas others utilize a table-based methodology. Moreover, many threat models cater to a very specific use case or a narrow field, making them even harder to compare. One thing that many models do have in common, however, is the perspective whence they approach assessing risk. More often than not, a risk is viewed from the perspective of the defender, i.e. the asset or service that is under attack. This then results in a model that mostly considers properties that are directly connected to the defender, such as the cost of defensive measures. It can be argued that this creates an incomplete model and that incorporating some form of attacker profiling can improve the overall quality of the model. The reasoning behind this is that the types of attacks an attacker can perform highly depend on properties such as his financial means, goals, and level of skill. By identifying the types of attackers a system will likely not have to deal with, some attacks may immediately become redundant to consider. This can make the threat model significantly less complex, as well as allow for critical resources to be deployed where they matter the most.

The threat modeling framework for which we propose an addition combines STRIDE and DREAD-D. STRIDE was developed at Microsoft for identifying information security threats. The name "STRIDE" is a mnemonic for security threats in six categories (**S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial-of-Service, and **E**levation of privilege). STRIDE has been used a lot in security research [10, 11, 12, 13] and has become one of the de facto standards for identifying threats. After identifying the risks with STRIDE, they are ranked using the DREAD risk assessment model. DREAD stands for **D**amage, **R**eproducibility, **E**xploitability, **A**ffected users and **D**iscoverability. We have decided to omit the final category of DREAD, as it encourages 'security through obscurity'. This modified model is more widely known as DREAD-D. In DREAD-D, risks are ranked by assigning a risk score between 0 and 10 to each of the four categories for every identified risk, with the final (omitted) category always receiving the maximum score. The risk with the highest total risk score receives the highest ranking. Due to their simplistic nature, STRIDE and DREAD-D are often used in conjunction with each other. Preserving this simplistic nature was one of our main goals in designing an addition to the modeling frameworks.

3.1 Attacker Profiling Model

As was briefly mentioned earlier, we propose an addition to the combination of STRIDE and DREAD-D. The reasoning behind this is that currently, STRIDE and DREAD-D do not consider the capabilities and shortcomings of attackers in any way. We argue that the inclusion of an attacker profile will lead to a threat model that is less complex, yet more accurate. In order to include an attacker profile in the framework, there first needs to be a

proper model to profile them. Profiling is the act of extracting unique characteristics from a person to build a profile that accurately describes said person's properties. One of the great benefits of building profiles of different personas is the ability to compare them in an objective way. Instead of relying on qualitative metrics, a person's unique traits are reduced to a quantifiable set of properties, such that the profile can then be equated to other profiles. The quality of a profiling model highly depends on how well it is able to represent a real person. However, adding more properties to a model does not necessarily make it more accurate. The more generalized a profiling model is, the easier it is to compare different profiles. Conversely, a model that is too generalized may fail to capture the intricate details that make a person unique. It should be clear that including the correct properties into a model for attacker profiling is key.

The methodology we used to design our model addition looks as follows. First, we performed a background study to get an overview of how much research has been conducted in the field of attacker profiling. During this study, we found a number of papers in which attacker profiling was touched upon. We then extracted the profile metrics that these studies used in their models or assessments, and combined and generalized them. Some metrics had to be discarded to keep the model quantifiable, as that property is required to directly compare attacker profiles. Finally, we designed a method to incorporate the attacker profile in the process of identifying and ranking risks. The following sections will elaborate on the details and results of the steps of this methodology.

3.1.1 Background

During our preliminary background study, we found a limited number of research papers in which attacker profiling was described or performed. We believe this is an indication of the state of research into attacker profiling. For each of the papers that we did find, we provide a brief summary of its relevance for the topic of attacker profiling. Furthermore, we extract all potential profile metrics that each research mentions. Please note that this section does not contain any assessment of the individual metrics, as that is the next step of our methodology.

Lenin et al. In [14], the authors propose a risk assessment model based on the existing ApproxTree [15] model. Their main contribution is the addition of an attacker profile to said model, comparable to what this research aims to achieve. Among other things, they present formal definitions for an *attack suite* and an *attacker profile*. An attack suite is "*a set of elementary attacks which have been chosen by the attacker to be launched and used to try to achieve the attacker goal*". According to their definition, an attacker profile is a function which takes an attack suite as an input. If a certain attacker is capable of performing all elementary attacks in an attack suite, the profile satisfies the suite.

The authors define the following profile metrics and descriptions.

- Budget - the monetary resources of the attacker, measured in currency units.
- Proficiency - the skill level of an attacker, measured on an ordinal scale (Low/Medium/High).

- Time - the available time resource of the attacker, measured on an ordinal scale (Seconds/Minutes/Hours/Days).

Dantu et al. In [16], the authors formulate a mechanism to estimate the risk level of resources based on attacker behavior. They utilize attack graphs to model all possible attack paths to a resource and then compute the overall risk level using a Bayesian estimation technique. To construct the actual attacker profiles they have used, the authors performed a survey. Unfortunately, the website on which this survey and its results are hosted is no longer online. As there is no explicit mention of which attacker properties were used for the construction of these profiles (except for references to the non-existent survey), the only information this research provides is a list of potential attacker attributes on a "for example" basis. However, the research does briefly discuss how certain ordinal values can be quantified by using a simple distribution between 0 and 1.

The authors mention the following profile metrics.

- Cost
- Computer and hacking skills
- Time
- Attitude
- Tenacity
- Perseverance
- Motives

Cardenas et al. In [17], the authors provide a novel, holistic view on security requirements and threat models in sensor networks. Furthermore, they present a ranking of threats and security mechanisms. They base their ranking on the difficulty of performing a certain attack. The difficulty of an attack is determined by considering a set of properties that bear close resemblance to the attacker properties we are after.

The authors define the following profile metrics and descriptions.

- Cost of extra hardware - does an attack require more than commodity hardware or is no extra hardware required?
- Physical access - refers to physical proximity to the sensors and being able to touch them.

- Required technical skill - how well can the attacker perform brute-force attacks and/or logical attacks[3]? Does he perform them manually or automated?

Due to the slightly different nature of the research, the descriptions of the metrics above have been transcribed to fit our research.

Filippoupolitis et al. In [18], the authors propose a system that is supposedly capable of both real-time profiling of human attackers and detecting bots. Although bot detection is not relevant to our research, their work regarding real-time attacker profiling is. As part of their research, the authors identified a list of properties of a potential cyber-attacker. The properties are related to the cultural, professional, and demographical characteristics of the person behind an attack. The list of properties was constructed with a real-time attack in mind and therefore contains metrics that are only relevant as an attack is taking place. Other metrics are more focused on the forensical evidence that an attacker leaves behind. However, for the sake of completeness, the metrics below are listed exactly as they are described in their paper.

The authors define the following profile metrics and descriptions.

- Skill - this feature captures the competence of the attacker in terms of IT skills.
- Education - the education level of this attacker is described by this feature.
- Risk - a person trying to avoid risk is very likely to behave in a different way compared to one that is risk prone. This feature tries to capture this aspect of the attacker's profile.
- Gender - being able to state whether the person behind the attack is male or female can significantly help a forensics investigation.
- Goal - this feature describes what was the reason for the attacker to target the computer system in question. A person that attacks a system to achieve financial gain needs to be treated separately compared to one that does it out of curiosity or to state his political beliefs.
- Speed - this feature measures the speed of the cyber attacker in commands per second. It is directly related to the attacker's IT skill level.
- Mistakes - a highly skilled attacker would make fewer mistakes compared to an inexperienced one. This feature captures the number of mistaken commands issued by the attacker during the cyber attack.
- Anti-forensics - one of the most important elements of a forensic investigation is the analysis of potential tracks the attacker has left behind. This feature discriminates

[3]The authors define a logical attack as an attack requiring specific knowledge about the protocol used by the network.

between attackers that tried to cover their tracks (e.g., by deleting log files) and the ones that did not take any such actions.

- Success - this feature describes whether the attacker was successful in carrying out the cyber-attack.

3.1.2 Metric Selection

The next step in our methodology is the selection of metrics we include in our attacker profiling model. We do this by first generalizing the different metrics we extracted from other papers, such that they all use the same terminology. Specifically, metrics related to *cost* are renamed to **Budget**, metrics related to *technical skills* are renamed to **Skills** and metrics related to *risk attitude* are renamed to **Risk**. We carefully assess which metrics are suited to be included in the attacker profiling model. This assessment is done based on two properties: *relevance* and *quantifiability*. The relevance of a metric indicates the amount of information that makes an attacker unique it contains. The quantifiability of a metric indicates how well the metric suits the (mostly) quantitative DREAD-D ranking model. For the first property we use an ordinal scale (Low/Medium/High) and for the second property we use a simple binary score (Yes/No). The actual metric selection is performed by applying the following rule to each metric m.

$$Accept_m = \begin{cases} \text{Yes} & \text{if } m_{relevance} = \text{High}, \\ \text{Yes} & \text{if } m_{relevance} = \text{Medium and } m_{quantifiable} = \text{Yes}, \\ \text{No} & \text{if } m_{relevance} = \text{Low} \end{cases} \tag{1}$$

The *relevance* property is leading in this selection process, as we believe that a highly relevant but not quantifiable metric should be included over one that is quantifiable but not relevant. The *quantifiability* property is mainly used to decide whether to select metrics of medium relevance.

Table 1 shows the list of selected metrics, including a short description and a measurement scale for each metric. The exact selection process is elaborated on in Appendix A.1. For metrics marked as quantifiable, the table also shows what scale of measure should be used for assigning values.

3.1.3 Model Integration

The final step in designing an attacker profile addition for STRIDE and DREAD-D is the actual integration of said addition into the framework. We initially set the goal to keep the proposed addition as quantitative as possible, as ranking with DREAD-D is performed on a quantitative basis. During the process of seeking, extracting, and selecting metrics for the attacker profiling model, we found that it is incredibly difficult to completely quantify a profile without losing valuable data. An attacker profile should not be seen as a single point in a 1-dimensional space, where different points in that space can be directly compared.

Metric	Description	Scale of measure
Budget	The monetary resources of an attacker.	Interval
Skills	The skill level of an attacker.	Ordinal
Time	The amount of time an attacker is willing to invest in an attack.	Ordinal
Physical access	Indicates whether an attacker can physically access the system he is attacking during the attack.	Ordinal
Risk	The level of risk an attacker is willing to take.	Ordinal
Perseverance	The perseverance of an attacker.	Ordinal
Education	The level of education that an attacker has.	Ordinal
Motives	The underlying reason an attacker has for attacking a system.	Qualitative
Goal	The goal an attacker needs to reach to consider his attack successful.	Qualitative

Table 1: The final list of metrics as selected to be included in the attacker profiling model.

Instead, every included metric adds a new dimension to the attacker profile. Direct comparison is possible as long as qualitative properties are excluded, as they make the space infinitely more complex. Furthermore, purely from the perspective of an attacker's threat level, his motives and goals seem to add little significant information. This, combined with the difficulty of designing a suitable scoring mechanism, ultimately led to the decision to make the model fully quantified. However, because the **Motive** and **Goal** metrics passed the selection process, they will instead be used for describing the attacker. Qualitative metrics help in illustrating what kind of person an attacker is, which in turn makes it easier to estimate the scores for other metrics. They also help eliminating attacks that an attacker is unlikely to perform. Although this is not entirely in line with the ideas we initially had, we believe this approach is closest to our original goal: keeping the model simplistic, yet accurate.

The attacker model is purely focused on the quantitative attributes within the attacker profile. We have defined a concrete scale of measure for every metric. These scales and their descriptions are shown in Table 2 and Table 3. It should be noted that not all scales are linear and not all scales are divided into ten equal parts. To compute the total threat level of an attacker, first estimate the value of each of the seven metrics for that specific attacker. Then, use the following equation to determine the overall threat level.

$$Threat_{attacker} = \frac{1}{|M|} \sum_{m \in M} S(m), \tag{2}$$

where M is the set of metrics in the model and $S(m)$ is the score that metric m was given. The result of this is a simple, basic mapping of seven attacker metrics to a single number,

Metric	Scale
Budget	Budget b in currency ¤ between: 1. 0.00 and 9.99 2. 10.00 and 49.99 3. 50.00 and 99.99 4. 100.00 and 499.99 5. 500.00 and 999.99 6. $1,000.00$ and $9,999.99$ 7. $10,000.00$ and $99,999.99$ 8. $100,000.00$ and $999,999.99$ 9. $1,000,000.00$ and $9,999,999.99$ 10. $> 10,000,000.00$
Skills	The attacker has the following computer and hacking skills: 1. Computer illiterate, no hacking skills. 3. Novice computer user, no hacking skills. 5. Advanced computer user, basic hacking skills using scripts and software found online. 7. Expert computer user, advanced hacking skills using self-written script and software. 10. Expert computer user, expert hacking skills with a deep knowledge of computer networks and systems.
Time	An attack should be completed within the order of: 1. Minutes 3. Hours 5. Days 7. Weeks 9. Months 10. Years

Table 2: The concrete scales of measure for the first three selected metrics. Not all scales contain ten discrete values, but scores between two discrete values are allowed. For instance, an attacker with a skill level somewhere between 3 and 5 may also receive a skill score of 4.

allowing attacker profiles to be compared directly. It also provides a convenient method to integrate attacker properties into DREAD-D. DREAD-D ranks threats by computing the average risk level from the values that are assigned to each of its categories, very similar to how we compute the overall attacker threat level in Equation (2). This threat level can be incorporated into DREAD-D by simply including it in the set of categories to compute the average of. The other DREAD-D categories all have values between 0 and 10, which also holds for our attacker profile due to the way in which the scales of measure were designed.

Metric	Scale
Physical access	Does the attacker have physical access to the system or network he is attacking? 0. No. 5. Yes, to a copy of the system to be attacked. 10. Yes, to the system to be attacked itself.
Risk	How much risk is an attacker willing to take? 1. None, he does not break any laws and never risks his own health or safety. 3. A little, he might do some things that can be considered illegal, but never risks his own freedom. 5. A moderate amount, he routinely breaks laws during his attacks, but prefers to not put his freedom at risk. 8. A high amount, he routinely operates illegally during his attacks, sometimes even risking his own health, safety and freedom in the process. 10. An extreme amount, he exclusively operates illegally during his attacks and constantly puts his health, safety and freedom at risk.
Perseverance	How perseverant is an attacker? 1. Not at all, he stops his attack after only a single failed try. 4. A little, he tries a few attacks using the same attack vector, but stops after they fail. 7. Very, he tries different attacks using different attack vectors. 10. Extremely, he tries (and retries) many different attacks using different attack vectors and heavily tweaks his attacks based on how they perform.
Education	What is the attacker's level of education? 1. Unschooled. 4. Low level of education. 7. Middle level of education. 10. High level of education.

Table 3: The concrete scales of measure for the last four selected metrics. Not all scales contain ten discrete values, but scores between two discrete values are allowed. For instance, an attacker with a risk level somewhere between 3 and 5 may also receive a risk score of 4.

3.2 Model Application

In this section, we apply the attacker profiling model we designed in the previous section. We then use STRIDE to identify the threats that a DMS network might have to handle at some point. Finally, we apply DREAD-D to rank the threats. The result of these three steps is a comprehensive but accurate list of threats that exist within a DMS network. This list serves as a starting point for designing a solution to secure data flowing through a DMS network.

Metric	Score
Budget	5
Skills	10
Time	7
Physical access	1.67
Risk	7.33
Perseverance	8
Education	8
Overall threat score	6.71

Table 4: The metric scores and overall threat level for the Cyber Criminal profile.

3.2.1 Attacker Profiles

As was established earlier, proper profiling of potential attackers is the foundation of simple yet accurate threat management. We have constructed three attacker profiles: **Cyber Criminal**, **Service Competitor** and **Animal Rights Activist**. For each profile, we estimated the values for the attributes that we included in our attacker profiling model. We gathered the information required to build these profiles through open conversations with three DMS developers. They all work on different technical parts of the DMS and thus each of them offers unique insights into the risks and threats they foresee.

Cyber Criminal The Cyber Criminal profile envelops what most people think of when they hear the word "hacker": an individual that is highly skilled in using computers and digital networks to attack or break into systems. In many aspects, he is comparable to a regular (non-cyber) criminal. His main motive is related to generating excessive income and he is not afraid to take risks to achieve his goals. He belongs to the top of his field and is prepared to spend weeks preparing a single attack. The largest limitations he faces are that he exclusively works online and operates on a limited budget. The complete scoring of this profile is shown in Table 4.

In general, the Cyber Criminal is likely to perform attacks that require proper planning and a very high level of technical knowledge. However, attacks that demand a large amount of money or physical access are unlikely to be executed by this attacker.

Service Competitor In this context, a Service Competitor is (someone representing) a company that competes with the DMS on a hardware and/or software level. An example of this would be a company that is developing a similar analysis tool. To save on costs, they might decide to build their tool such that it is compatible with the DMS we are considering in this research. This allows them to skip the process of designing, developing, and producing hardware devices by making use of the resources of a competitor. Another possible motive for thoroughly investigating a deployed DMS network (and thus attacking it) is to compare it with their own. This could allow them to take factors that make this DMS unique and copy them for their own use. This, too, can save them significant sums of money on development costs.

Metric	Score
Budget	7.67
Skills	7.33
Time	9
Physical access	4.67
Risk	4
Perseverance	7.67
Education	8.67
Overall threat score	7.00

Table 5: The metric scores and overall threat level for the Service Competitor profile.

The Service Competitor profile scores high on most metrics. They spend a lot of resources on research and development and can thus reserve a considerable amount of time and money for trying to gather information on a competitor's technology. In terms of skill, the Service Competitor receives an average score. The reasoning behind this is that a legitimate company in this field is unlikely to hire hacking professionals, as that could significantly damage their reputation. For that same reason, they are unlikely to take large risks. The complete scoring of this profile is shown in Table 5.

Animal Rights Activist The Animal Rights Activist is an attacker with a single motive: protecting animal well-being. The activist is characterized by his seemingly infinite perseverance and high willingness to take risks. Organizations such as Greenpeace are known to perform dangerous actions, like physically pursuing whaling fleets to stop their activities[4]. In some countries, large corporations have vertically integrated meat supply chains. This means that if such a company violates certain animal rights in one aspect of the supply chain, activists can directly hurt the company by attacking their deployed dairy management systems. The fact that this can damage a supply chain that potentially feeds thousands of people does not concern the activists too much. The Animal Rights Activist has a medium-sized budget for setting up cyber attacks and is also decently skilled in performing them.

The Animal Rights Activist is likely to infiltrate a deployed DMS network physically. This, combined with his perseverance and willingness to take risks, makes him a reckless and potentially dangerous attacker. The activist's skill level inhibits him from performing very complex attacks, but with the help of publicly available tools he is able to disrupt a farm's operations significantly. Furthermore, information they are able to obtain through the DMS might lead them to physically infiltrate a farm, something that other attackers are often not willing to do. The complete scoring of this profile is shown in Table 6.

3.2.2 Threat Identification & Ranking

The next step in the threat assessment process is the identification of threats. This is a critical step, as knowing what you are protecting yourself against is key in designing a

[4]https://www.reuters.com/article/environment-whaling-greenpeace-dc/greenpeace-ship-chases-japanese-whaling-fleet-idUSSYD32502420080112

Metric	Score
Budget	5.67
Skills	6
Time	7.67
Physical access	7
Risk	7.67
Perseverance	7.67
Education	6.67
Overall threat score	6.91

Table 6: The metric scores and overall threat level for the Animal Rights Activist profile.

proper security solution. We have targeted threats that arise from the lower levels of the system. This means that threats specific to high-level details such as hardware or operating system are omitted. Our focus lies on directly protecting the data flowing through the system and ensuring that it remains secure in the worst scenarios. Furthermore, threats related to social engineering are also not considered here. The cyber security community has long identified that "*security is only as good as its weakest link, and people are the weakest link in the chain*" [19]. Taking all possible human errors into consideration would make it impossible to construct a practical solution. Our final remark is the omission of the *Repudiation* category in the application of STRIDE. Repudiation is often achieved by deleting or modifying log files after an attack has occurred. This generally happens on a high level of a system or network, such as its file system. Due to our focus on threats that manifest in the lower levels, we believe this is outside of the scope of our research.

Risk identification requires a deep knowledge of the system being protected, as well as the environment it resides in. To gain more insight regarding the vulnerabilities the DMS faces when it is deployed, we once again conducted a series of interviews with DMS developers. The interviews again had the form of open conversations. We first briefly explained the three attacker profiles that were constructed earlier, and then discussed the potential impact of those attackers on the DMS. Using the information we gained during these interviews, we construct a list of threats that are both realistic and dangerous, and determine to which categories of STRIDE each threat belongs. This same approach to applying STRIDE has been used in [12]. For each threat, we then determine which attackers have the capabilities to perform it. Finally, we compute the risk score using DREAD-D in combination with our attacker profiles. For threats that can be performed by multiple attackers, we calculate the risk score for each of those attackers. Table 7 shows the list of threats that we compiled.

There are a few things to learn from the threats in Table 7. The DMS currently has no support for network-level authentication. As a result, anyone can enter the DMS network and pretend to be a legitimate device. Encryption can help solve this issue, because illegitimate senders do not have the information required to construct valid encrypted messages. Furthermore, incorporating some form of access control can also be achieved with encryption. A device's secret key can implicitly contain information about its role in the network, making it very hard for adversaries to elevate the privilege of compromised devices.

	Threat	S	T	I	D	E
1	Sending instructions to devices by pretending to be a PU.	✓	✓			✓
2	Skewing statistics by inserting data through a fake Animal Tracker.	✓	✓			
3	Skewing statistics by inserting data through a fake repeater.		✓			
4	Disrupting system operations by jamming radio signals.				✓	
5	Disrupting system operations by overflowing devices with IP packets.				✓	
6	Eavesdropping on animal-specific information.			✓		
7	Eavesdropping on system-related information.			✓		
8	Gaining access to data that is normally not accessible due to license restrictions.			✓		

Table 7: The list of threats that were identified, including the STRIDE categories that they belong to.

After threat identification, we once again conducted interviews with DMS developers to rate each of the threats on the four DREAD-D categories. Table 8 shows how each threat has been scored on the different categories. Recall that DREAD-D stands for **D**amage, **R**eproducibility, **E**xploitability, **A**ffected users, and **D**iscoverability, where the **D**iscoverability always receives the maximum score of 10. This scoring also includes our novel attacker threat level. Attacks that can be performed by multiple attackers appear in the table multiple times, once for each attacker. From the ordering of this list, we can see that attacks related to *data confidentiality* and *data integrity* have the potential to cause the greatest damage when performed by their respective capable attackers. We have carefully addressed these issues in our proposed solution in an attempt to provide proper data security against these threats.

3.3 Discussion

In this section, we designed and applied a model to uniquely profile potential attackers of a system. The design of this model is based on existing research and literature on attacker profiling in IT systems. The exact details of the model, including their incorporation in STRIDE + DREAD-D, are novel. The model was applied using the insights and opinions of experts from different fields. During our discussions with these experts, as well as during our own work with the model, a number of shortcomings surfaced. In this section, we will address these shortcomings and discuss potential solutions.

Redundancy of Education metric During the process of selecting the relevant metrics from the list of extracted metrics, we wrote that the **Education** metric was of medium relevance. We hypothesized that an attacker's level of education influenced other properties, such as his technical skill level. Furthermore, we assumed that a high level of education relates to having a well thought out plan, making a highly educated attacker more dangerous than a lowly educated one. As was discovered during the interview sessions, these assumptions were not completely true. Estimating a threat score based on a person's level

Threat	Attacker	D	R	E	A [1]	D	Att. threat	Threat score
6	Animal Rights Activist	6	8	9	8	10	6.91	7.99
7	Service Competitor	8	8	9	3	10	7	7.50
7	Cyber Criminal	8	8	9	3	10	6.71	7.45
1	Cyber Criminal	8	8	9	1	10	6.71	7.12
4	Animal Rights Activist	4	7	7	1	10	6.91	5.99
2	Service Competitor	8	8	1	1	10	7	5.83
3	Service Competitor	8	8	1	1	10	7	5.83
5	Animal Rights Activist	4	7	3	1	10	6.91	5.32
5	Cyber Criminal	4	7	3	1	10	6.71	5.29
8	Service Competitor	8	5 [2]	2	3	10	7	5.17

Table 8: A breakdown of the DREAD-D scoring that each threat received. This scoring also includes our novel attacker threat level. The table has been ordered by overall threat level. **[1]**: Considering the limited amount of users involved in this system, a score of 1 indicates that only a single farmer is affected, a score of 3 indicates that the DMS developer is affected, and a score of 8 indicates that multiple farmers are affected. **[2]**: This attack requires a lot of variable data related to licenses, making an automated attack difficult.

of education produced inaccurate results. An attacker might have no formal education, yet have gained a very deep knowledge of computer networks and systems through self-study work. Moreover, the **Skill** and **Education** metrics cover essentially the same properties, namely the amount of knowledge an attacker possesses. For these reasons, it can be argued that the **Education** metric should be removed from the model.

No weighting factors In its current form, our model does not support weighting factors for the different metrics. During the design process, the introduction of weighting factors was discussed briefly. However, due to a lack of knowledge required to assign concrete numbers to those weights, we ultimately decided to not use such coefficients. The result of this decision is that all metrics weigh equally as much in the overall threat level computation. For instance, the increase in overall threat level is the same when an attacker increases his budget from ¤5.00 to ¤100.00, and when he increases his skill level from a novice computer user to an expert computer user (a total of $\frac{4}{7}$ in both cases). It is clear that the latter makes the attacker a far more dangerous adversary; something that should be reflected in the overall threat level. The obvious solution to this problem is to introduce weighting factors to make some metrics more impactful than others, but this requires more insight into the distribution of weights.

Non-linear scales For some metrics, we initially decided to use a scale that is nonlinear. Our reasoning behind this decision was that for some metrics it is simply not feasible to design a 10-point scale. Unbounded metrics, such as **Budget**, do not have an upper limit. We solved this by setting an artificial upper limit of ¤10,000,000.00, meaning there is no differentiation between budgets higher than that. However, that still leaves nine intervals to be set up. If we had chosen a linear scale, then every attacker with a budget of less than approximately ¤1,000,000.00 would receive a score of 1 on this metric. Looking at

the scores we have given our profiles, the highest **Budget** score among the three would be a 2. Clearly, a linear scale is not always the best method. However, the clear downside to our current **Budget** scale is that a small attack budget increase can lift the metric's score quite significantly. An increase of e.g. ¤1,000.00 can raise the score of this metric from 1 to 6, leading to an increase of almost a full point in the overall threat level. It can be argued that percentage-wise, this small budget is as valuable to a low-budget attacker as a much higher budget increase is to a high-budget attacker. A possible solution for this skewed distribution of metric importance could again be the use of weighting factors. Another option is removing the fixed 10-point scale, although this would require further tweaking of the model. The 10-point scale was chosen because it is also used in DREAD-D and makes the integration of our model with that framework easier.

4 Symmetric Cipher Evaluation

In the second part of this research, we develop and perform a series of benchmarks to determine which type of encryption is most suited for this IoT-like network. Like IoT devices, Animal Trackers operate with very limited energy resources. Currently, an Animal Tracker is able to run for approximately ten years without needing a battery replacement. This battery life is a hard requirement for the system. Animal Trackers have to be water-proof and shock-proof to survive years of constant usage on a farm. Battery replacement is practically impossible and thus all components have only limited energy resources to work with. The microcontroller unit (MCU) that was chosen for this research is the EFR32MG22 Series 2 Zigbee SoC[5]. It is manufactured by Silicon Labs and features an ARM Cortex M33 core and low-power peripherals. It is part of the Wireless Gecko Series 2 platform, which is focused on delivering high-performance wireless devices at a very low power cost. When deployed in the field, the microcontroller's operations are highly optimized to keep power consumption as low as possible. To simulate this efficiency in our experiments, we set the clock speed of the MCU's Cortex M33 core to a static 20 MHz using FSRCO[6] as the clock generation source. The exact core clock speed that will be used in production depends on how efficient the MCU is able to operate at different speeds. Determining this value is outside the scope of this research and the usage of the FSRCO only allows a clock speed of 20 MHz, hence the decision to set it to that speed. The fast start-up property of the FSRCO has the additional benefit that little time and energy is wasted because of lengthy core start-up. During our initial testing, we used a different clock source and noticed a significant increase in both time and energy consumption. The usage of the FSRCO eliminates this issue.

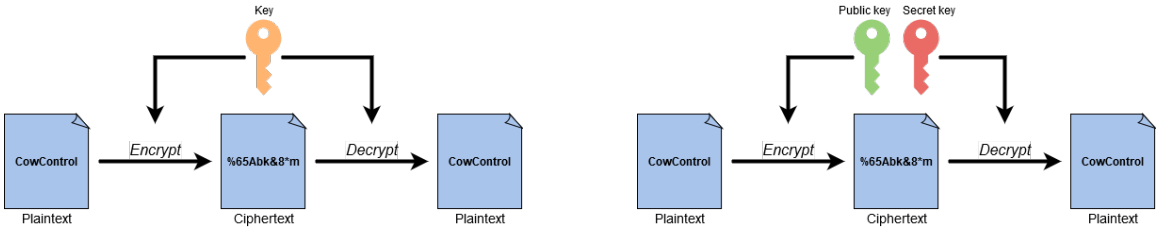
4.1 Background

The second part of our research revolves around three topics: ciphers, modes of operation, and authenticated encryption. In this section, we briefly introduce all three and provide the information necessary to understand the decisions we have made.

4.1.1 Ciphers

In cryptography, a *cipher* is an algorithm for performing encryption and decryption. Well-known examples of ciphers are `AES` [20] and `RSA` [21]. Modern ciphers can be divided into two main categories. The first category are the *symmetric* ciphers. A symmetric cipher utilizes a single key, which it uses for both the encryption and decryption operation. Symmetric ciphers are further subdivided into two categories: block ciphers and stream ciphers. Block ciphers operate on a fixed number of input bits, called the *block size*. Input data that is not a multiple of a cipher's block size must be padded to that length first. In stream ciphers, the input data is combined with a pseudorandom stream of bits. As a result, stream ciphers can operate on input data of any size. The second main category are the *asymmetric* ciphers. An asymmetric cipher uses two keys: one for encryption and one for decryption. Asymmetric ciphers are often referred to as *public-key* ciphers, because the encryption key

[5]More specifically, the EFR32MG22C224F512IM40.
[6]Fast Start-up RC Oscillator



(a) Symmetric cryptography **(b)** Asymmetric cryptography

Figure 1: The difference between symmetric and asymmetric cryptography. In symmetric cryptography, the same key is used for both encryption and decryption. In asymmetric cryptography, two different keys are used for encryption and decryption. The encryption key is often made public, such that any sender can encrypt data. Only the receiver, who holds the secret key, is able to decrypt that encrypted data.

can be made public. This public key allows any sender to encrypt data. The decryption key, also called the secret key, is kept secret by the receiving party. This *many-to-one* property allows multiple senders to securely communicate with a single receiver. Figure 1 shows a simplified representation of both symmetric and asymmetric encryption.

Both types of ciphers have their advantages and disadvantages. Symmetric cryptography relies on a piece of secret information that both communicating parties share. This shared secret has to be established in some way, so that the parties can derive the key that they have to use to encrypt and decrypt data. This process is called *key agreement* and comes with many security considerations of its own. In the third and final part of this research, we take a closer look at key agreement and its difficulties. The benefit of using symmetric cryptography over asymmetric cryptography is its efficiency and speed. Asymmetric cryptography generally relies on certain mathematical properties that make it possible to encrypt with a public key that discloses no information about the secret key. Albeit useful, these properties make the encryption of data a computationally intensive process. Symmetric ciphers do not require such constructs, as they require only a single key. This makes them faster and far more energy efficient.

In practice, most applications use a so-called *hybrid cryptosystem*. In a hybrid cryptosystem, asymmetric cryptography is used to agree upon a key that will be used for symmetric cryptography. In other words, the best properties of both symmetric and asymmetric cryptography are combined. It provides the convenient *many-to-one* property of public-key encryption, with the speed and efficiency of symmetric encryption. Furthermore, if both schemes are secure, the resulting hybrid cryptosystem is secure, too, as has been proven by Kramer and Shoup [22]. In their recent work, they show that a combination of an appropriately secure *key encapsulation mechanism* (a public-key encryption scheme, in this context) and an appropriately secure symmetric encryption scheme yields a secure public-key encryption scheme. The exact application of this principle will be discussed in the next and final part of this research, but for now we simply assume that symmetric cryptography will be used for the majority of the encryption operations.

Six different ciphers have been used in this part of the research. These ciphers are `AES`, `ARIA` [23], `Blowfish` [24], `Camellia` [25], `ChaCha20` [26], and `XTEA` [27]. `AES`, likely the

best known cipher in this list, is based on a substitution-permutation network (SPN). An SPN uses a series of substitutions and permutations in combination with a key to scramble input data such that it is no longer distinguishable from random data. `ARIA`, like `AES`, also uses an SPN to convert a plaintext into a ciphertext. `Blowfish`, `Camellia`, and `XTEA` are Feistel ciphers (although `Blowfish` and `Camellia` also utilize some SPN techniques). A Feistel cipher applies a round function to input data multiple times in a row (called 'rounds') to produce a ciphertext. `ChaCha20` is a stream cipher that relies on a series of bitwise *add-rotate-XOR* (ARX) operations. ARX operations are relatively fast and do not require complex implementations like some SPNs or Feistel ciphers do. Two additional ciphers we have evaluated, that are not in the list we presented earlier, are `AES(HW)` and `ChaCha20-Poly1305`. `AES(HW)` is based on `AES`, but with hardware acceleration enabled. `ChaCha20-Poly1305` is a variant of `ChaCha20` that enables authenticated encryption, which we will discuss in Section 4.1.3.

Note that so far, we have only discussed encryption functions. The performance of a cipher's decryption falls outside the scope of this research. Whereas encryption is performed by the restricted Animal Tracker, decryption of messages is done on the PU. This device has significantly more power and computing resources available, so the energy efficiency of decryption is irrelevant.

4.1.2 Modes of Operation

Block ciphers operate on input data of a fixed length. This means that in order to encrypt a longer sequence of bits, the encryption function has to be applied multiple times. A *mode of operation* describes how this repeated encryption occurs in a secure fashion. Different modes of operation have different implications for security. In this section, we look at the seven modes of operation that we have used in our evaluation. For each mode, we present how it works and what its advantages and disadvantages are.

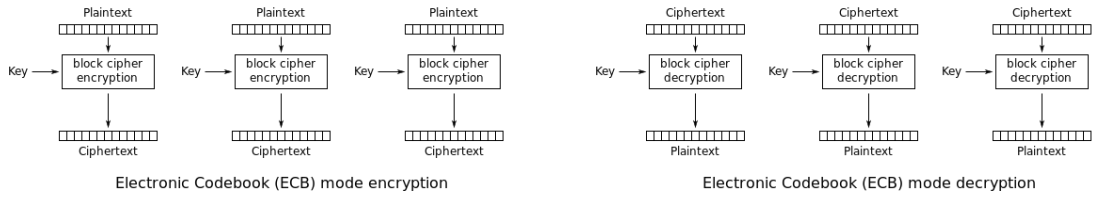


Figure 2: In `ECB`, every block of plaintext is encrypted or decrypted individually. Image source: [28]

ECB is the simplest of the encryption modes. `ECB` stands for Electronic Codebook, named after conventional physical codebooks. In `ECB`, the plaintext is divided into blocks and every block is encrypted individually. Figure 2 illustrates this process. A simple, yet effective attack against `ECB` involves inserting new (or existing) blocks of ciphertext into another ciphertext. Because there is no relation between the different blocks, the receiving

side will be able to decrypt the ciphertext without knowing it has been tampered with. Furthermore, `ECB` lacks diffusion, meaning that identical blocks of plaintext are encrypted to identical blocks of ciphertext. This makes data patterns in the plaintext stand out in the ciphertext, too. `ECB` should never be used on its own, as it provides little to no security features.

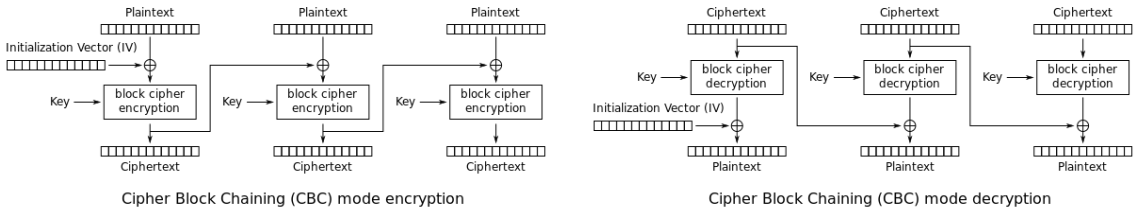


Figure 3: In `CBC`, every block of plaintext is XORed with the previous ciphertext before being encrypted. For the first block an initialization vector (IV) is used in the XOR operation. This IV provides semantic security: identical messages with different IVs produce different ciphertexts. Image source: [28]

CBC is an abbreviation for Cipher Block Chaining. As this name suggests, each block of encrypted ciphertext is used as input for the encryption of the next block, as shown in Figure 3. This chains the blocks together, offering more protection against attacks that rely on the insertion of data in ciphertext. However, because the blocks in `CBC` are only reliant on the plaintext and the ciphertext of the previous block, an attacker could potentially modify ciphertexts by simply truncating them. Another downside to `CBC` is the fact that encryption is not parallelizable and thus inherently less efficient than modes that are.

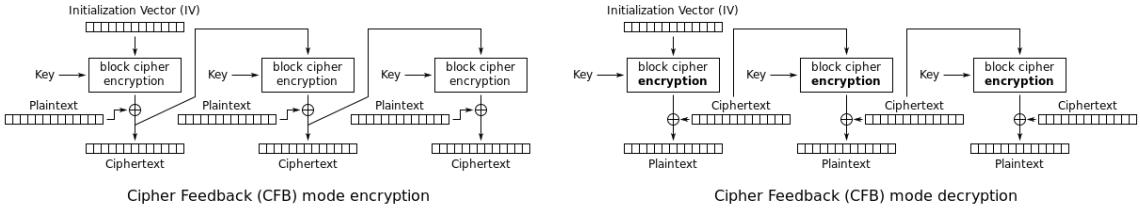


Figure 4: In `CFB`, the ciphertext (or the first s bits thereof) is encrypted and XORed with the plaintext, to produce a new block of ciphertext. Like `CBC`, `CFB` also uses an IV. Image source: [28]

CFB is short for Cipher Feedback Mode. In its simplest form, `CFB` encrypts the ciphertext of the previous block to generate a stream of bits in chunks, called a keystream. Encrypting plaintext is as simple as XORing it with the keystream, as is shown in Figure 4. This turns the underlying block cipher into a stream cipher. A more complicated variant of `CFB` uses truncated feedback. This variant produces a keystream in chunks and then truncates

the keystream chunk to s bits, where $1 <= s <= b$ for a b-bit block cipher. `CFB` variants with a low s value are used for their good error propagation properties. However, the clear downside to this is the fact that a keystream is generated in chunks of s bits instead of b bits. This makes it so that `CFB` is often one of the slowest and least efficient modes of operation.

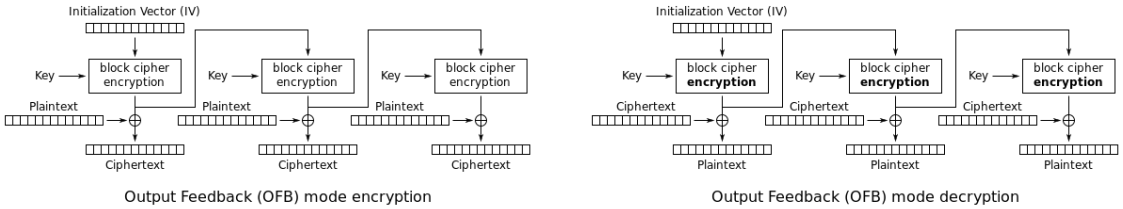


Figure 5: `OFB` encrypts a ciphertext to generate keystream. This keystream is then XORed with plaintext to generate a new block of ciphertext. Like `CFB`, `OFB` turns the underlying block into a stream cipher. Note that the encryption and decryption functions are identical. Image source: [28]

OFB stands for Output Feedback. `OFB` encrypts the ciphertext of the previous block to generate a keystream, turning the underlying block cipher into a stream cipher. As displayed in Figure 5, the algorithms for encryption and decryption with `OFB` are identical. This is caused by the symmetry of the XOR operation. Initially, `OFB` supported truncated feedback as used in `CFB`, but research has shown that this significantly weakens the security properties of the mode [29].

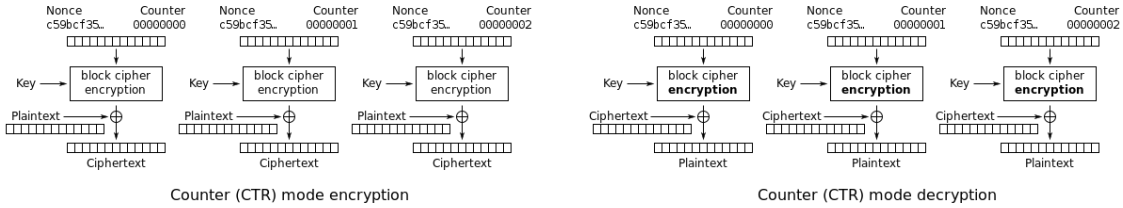


Figure 6: In `CTR` mode, an increasing counter is encrypted with the block cipher to create a keystream. This keystream is XORed with plaintext to form the ciphertext. Note that the encryption and decryption functions are identical. Image source: [28]

CTR is an abbreviation for Counter Mode. Like `CFB` and `OFB`, `CTR` generates a keystream by encrypting a nonce (IV) and successive values of a counter. In practice, a simple increment-by-one strategy is often employed for this counter. `CTR` is parallelizable in both encryption and decryption and allows random access during decryption. Especially the former gives it an edge in terms of performance over other modes. In terms of security, `CTR` may reveal information about plaintext if a (nonce, key) pair is ever reused for two different

messages. Furthermore, very precise malleability of ciphertexts is simple due to the lack of chaining or feedback. Bit flips in the ciphertext show up at the same index in the decrypted plaintext.

4.1.3 Authenticated Encryption

While most of the modes described in the previous section provide good security against attacks on confidentiality, data integrity is still an issue. A receiver will not be able to distinguish a carefully crafted ciphertext with fake information from a real message, because decryption will succeed regardless of the content of the data. In practice, to ensure that the decrypted data is equal to the original plantext data (in other words, the encrypted data has not been tampered with), message authentication is applied. This is generally done in the form of a Message Authentication Code (MAC): a short message tag that is computed by the sender using the plaintext and by the receiver using the decrypted data. The sender attaches his MAC and the receiver compares it to his. If they are equal, the receiver can be certain that the message has not been tampered with. Although this process of attaching a MAC to an encrypted message sounds intuitive and simple, its application comes with a number of practical considerations. For example, it is likely that both the encryption function and the MAC function require a key. Using the same key might lead to unwanted weaknesses, such as in the case where the encryption and MAC functions are very similar. Another consideration regards the fact that both functions likely require an IV. Again, using the same IV for both encryption and the MAC might cause weaknesses, especially since the security requirements for the two IVs may be different. *Authenticated encryption* schemes solve all these practical issues by combining encryption and message authentication under a single scheme, usually requiring only a single key and IV. This way, the user does not have to worry about the intricacies of both implementations and can rather focus on the overall design of his cryptosystem.

A slightly more elaborate variant of authenticated encryption is *authenticated encryption with associated data* (or AEAD). An AEAD scheme allows a sender to guarantee the integrity of both encrypted and unencrypted information. This type of scheme is often used in situations where a message header does not require confidentiality, whilst the payload does. Integrity-wise, both must be verified by the receiver, though. An alternative to using an AEAD scheme would be to simply include the header data in the encrypted payload. In a restricted environment, however, this increase in plaintext size might also increase the power consumption of the encryption operation.

We evaluated two modes of operation that offer authenticated encryption with associated data: `CCM` and `GCM`. In the rest of this section, we will briefly look at both modes and discuss their security implications.

CCM stands for Counter Mode with CBC-MAC. It uses `CTR` mode to encrypt data and guarantees data integrity with `CBC-MAC`, applied in an authenticate-then-encrypt fashion. This means that a message tag t is first computed with `CBC-MAC` and then both the plaintext data and the tag t are encrypted with `CBC-MAC`. This sequential structure means that the use of `CCM` over just `CTR` will likely cause a decrease in throughput. Whether authenticated

encryption is worth it at that cost depends on application-specific details. The security of CCM depends largely on the security features of the underlying cipher [30]. Furthermore, the aforementioned CTR issue that occurs when a (nonce, key) pair is used more than once also exists in this mode.

GCM is short for Galois/Counter Mode. Like CCM, GCM uses CTR for encryption. However, the generation of a MAC is performed by a series of mathematical operations in a Galois field, referred to as *GHASH*. Another difference compared to CCM is the fact that the MAC and the ciphertext are computed simultaneously, as shown in Figure 7. In CCM, the encryption and message authentication steps are performed sequentially. In GCM, this is done in parallel. This, combined with the use of CTR for encryption and fast Galois field multiplications for message authentication, makes GCM one of the best performing modes of operation that offer authenticated encryption [31]. However, the embedded computing world has voiced concerns regarding the incompatibility of GCM's parallelization with the cryptographic hardware accelerators on most microcontrollers[7].

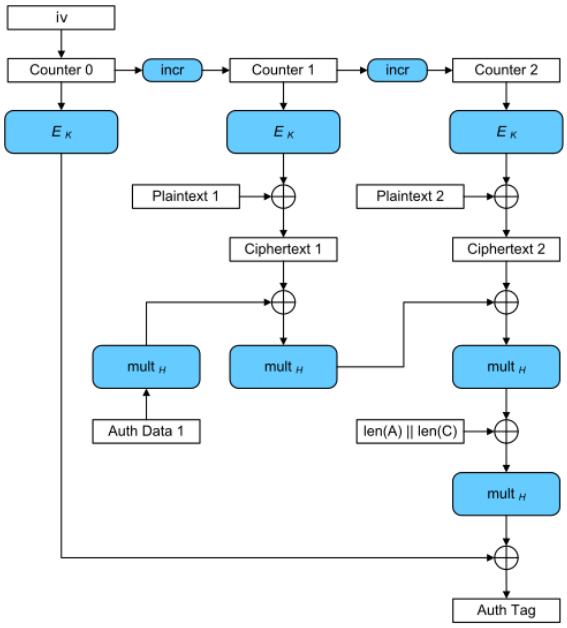


Figure 7: GCM combines CTR for encryption with a Galois hash ('mult' in the figure) for message authentication. The encryption and building of the MAC are performed in parallel. Image Image source: [28]

As was the case with CCM, the security of GCM depends on the underlying cipher, as the mode itself is provably secure under the concrete security model [31]. Nonetheless, it still requires a unique nonce for every message as long as the same key is used.

[7] https://www.silabs.com/community/blog.entry.html/2016/05/06/iot_security_part6-dNwC

4.2 Methodology

Before elaborating on the process of building, performing, and analyzing the cipher benchmark runs, we first list the specific tools that we have used.

IAR Embedded Workbench[8] was used to build and upload code to the MCU. IAR Embedded Workbench is developed by IAR Systems and contains a wide array of tools to assist in the development of embedded software. Especially the ability to directly view the MCU's memory proved useful on multiple occasions. IAR Embedded Workbench comes with an optional command-line interface, allowing a developer to write and debug code from other IDEs.

Simplicity Studio 5[9] was used to gather the energy consumption data during the benchmark runs. Simplicity Studio is developed by Silicon Labs, the manufacturer of the used MCU. It contains several features that are designed very specifically for their microcontrollers, such as the ability to monitor and save energy consumption data using their Energy Profiler tool. This tool was essential for this research, as this would otherwise require highly specialized hardware. The Energy Profiler can display and record real-time energy information, such as the device's average current and voltage. It is then able to export recorded data to either a CSV or an ISD file. One limitation of the Energy Profiler is that exporting data to a CSV file limits the sample rate to 100 Hz. Exporting to an ISD file, a seemingly proprietary filetype about which the company has shared no information, allows for sampling at a rate of 10 kHz. Fortunately, Silicon Labs kindly provided us with a Python script that is able to parse the ISD file and convert its contents into a CSV format. Interestingly, the contents of this Python script revealed that an ISD file is simply a compressed archive containing the actual samples.

Mbed TLS[10] is the C library we used to implement the encryption operations. The main reason for selecting this particular library was because Silicon Labs distributes their own version of it. In this version, they completely reimplement certain ciphers and modes of operation to be optimized for the hardware they manufacture. In the case of our EFR32MG22 MCU, it meant that the `AES` cipher and all of its available modes of operation received a hardware-accelerated implementation.

4.2.1 Design

Table 9 lists the ciphers whose performance we have evaluated in our benchmarks, including the different parameters they support. Most of this list was compiled based on which ciphers are supported by Silicon Labs' distribution of Mbed TLS. For every cipher, we have evaluated the performance of different modes of operation. Next to that, we also tested multiple combinations of keys and message sizes for each of the modes. Considering the ciphers support different key lengths and have different block sizes, the decision was made to test all ciphers with three keys and four messages. The keys had lengths of 128 bits,

[8] https://www.iar.com/iar-embedded-workbench/

[9] https://www.silabs.com/products/development-tools/software/simplicity-studio

[10] Part of the latest Gecko SDK as distributed with Simplicity Studio 5

Cipher	Modes	Keys
AES	CBC, CCM, CFB128, CFB8, CTR, ECB, GCM, OFB	128, 192, 256
ARIA	CBC, CCM, CFB128, CTR, ECB, GCM	128, 192, 256
Blowfish	CBC, CFB64, CTR, ECB	128, 192, 256, 448
Camellia	CBC, CCM, CFB128, CTR, ECB, GCM	128, 192, 256
ChaCha20	Stream	256
ChaCha20-Poly1305	Stream	256
XTEA	CBC, ECB	128

Table 9: The list of ciphers whose performance has been evaluated, including the different modes and keys they support. The different message lengths have been omitted from this table, as all ciphers support the message sizes of 128, 256, 512, and 1024 bits.

192 bits, and 256 bits and the messages had lengths of 128 bits, 256 bits, 512 bits, and 1024 bits. Only `Blowfish` received an additional 448-bit key, because it supports keys up to that size. Conversely, the two `ChaCha20` variants and `XTEA` were only tested with 256-bit and 128-bit keys, respectively, due to restrictions in their specifications. It should also be noted that `ChaCha20` and `ChaCha20-Poly1305` are stream ciphers and do therefore not require padding, potentially giving them an edge over those that do. However, selecting and evaluating the efficacy of different padding methods is outside the scope of this project. As such, the message sizes have been selected such that they fit in an integer number of blocks for all block ciphers.

Recall that the purpose of this second part of the research was to evaluate which symmetric encryption method is most suitable for use in this DMS. The definition of *most suitable* depends on different factors and variables. Therefore, the first step in performing these benchmarks was to identify which boxes a cipher has to tick to be considered as such. To assess this, we have defined three performance metrics by which we will compare ciphers:

1. The duration of a single encryption operation, measured in microseconds (μs),
2. The CPU usage of a single encryption operation, measured in CPU cycles,
3. The average electrical current a single encryption operation draws, measured in microamperes (μA).

The duration of a single operation gives an indication of how much time overhead the addition of encryption will introduce. Measuring the number of CPU cycles a single operation requires allows us to compute how many cycles a cipher needs to encrypt one byte of data. This makes it easier to compare ciphers with different parameters that are otherwise hard to directly compare. Finally, the amount of power drawn by a single operation helps us estimate the impact of encryption on the battery life of Animal Trackers. Additionally, hardware-accelerated ciphers run largely independent from the CPU, which means that simply counting CPU cycles does not accurately reflect the performance of said ciphers.

$$E_{(\mu Wh)} = U_{(V)} * I_{(\mu A)} * t_{(h)} \tag{3}$$

By combining the chip's voltage U, the average electrical current I, and an encryption operation's duration t with Equation (3), we can also compute how much energy every single encryption operation draws. This information will be used to determine exactly how much energy a cipher consumes over extended periods of time. Furthermore, because this metric combines two out of three metrics, we also occasionally use it as a fourth "summarization" performance metric.

4.2.2 Execution

The term 'cipher benchmark run' has been mentioned multiple times now. Let us first explain what we mean by this term. A cipher benchmark run is a series of encryptions where we encrypt a static message with a static key, using every mode of operation available to that cipher. `AES`, for instance, supports eight modes of operation: `CBC, CCM, CFB128, CFB8, CTR, ECB, GCM and OFB`. A single `AES` benchmark run would consist of eight encryptions (one for each mode) with the same key and the same message. Furthermore, every one of these cipher benchmark runs was repeated exactly fifty times. During preliminary data processing tests, we found that there was a lot of variance in the measured data. More specifically, two back-to-back measurements would sometimes show an increase of 100% in their data. Especially the measured duration of a single encryption seemed to vary a lot. This was likely a byproduct of our method of processing raw measurement data. More information about this process can be found in Section 4.2.3. By repeating every cipher benchmark run fifty times, we were able to smooth out these irregularities.

The three performance metrics we defined earlier were measured using different methods and tools. CPU usage was measured by using the `CYCCNT` register on the MCU. This memory register counts the number of CPU cycles that have passed since its last reset. We reset this number *before* every encryption operation and then read and stored its contents *after*. The duration and power consumption of the ciphers were measured using the Energy Profiler tool in Simplicity Studio. Figure 8 shows a screenshot of the Energy Profiler tool of one of our `AES` (with hardware acceleration) measurements. The actual measurement is indicated with a green line. During every cipher benchmark run, we measured three variables with a sample rate of 10 kHz: the relative time since the start of that benchmark run (in μs), the current electrical current (in mA) and the current voltage (in V). As can be seen in the figure, the voltage (orange) remains constant, whereas the electrical current fluctuates as the MCU performs encryption operations. Notice that the measurement data consists mainly of peaks in groups of two (two nonconsecutive sets of peaks have been indicated in the figure with red rectangles). These peaks were created deliberately to generate visual indications in the energy graph of when encryption starts and ends. We achieved this by letting the MCU enter a deep sleep state for 200 milliseconds *before* and for 100 milliseconds *after* every encryption operation. During deep sleep, the power consumption of the MCU drops significantly, creating a visual "dip" in the graph. This approach allowed us to visually identify how a cipher was performing and apply automated tools to process the raw data. Note that only the first peak in every pair is of interest to us. The second peak is caused by our code validating the encryption output through decryption. The data from these secondary peaks was not used in our analysis and was filtered out at a later stage.

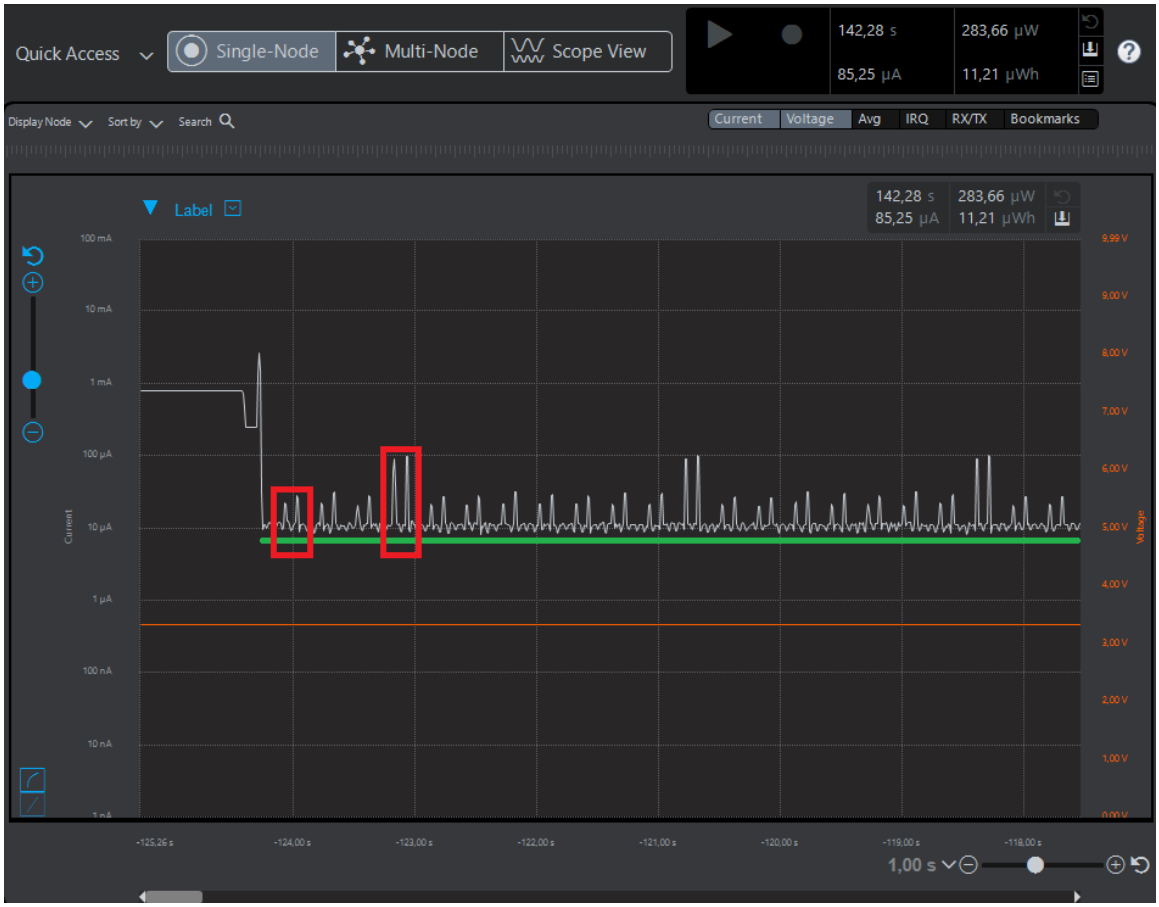


Figure 8: A screenshot from the Energy Profiler tool in Simplicity Studio showing an `AES` (with hardware acceleration) measurement. The green underlined part of the graph indicates the actual measurement, the rest is initialization of the chip. Two non-consecutive sets of peaks have been marked with a red rectangle.

Benchmark runs were performed for every cipher in Table 9, including a second run for `AES` with hardware-acceleration turned on (denoted as `AES(HW)`) in the rest of this paper.

4.2.3 Data Processing

After a benchmark run was finished, we used the script provided by Silicon Labs to convert the resulting ISD file into a CSV file. Most of the resulting CSV files contained over one million data samples, so to ensure efficiency and consistency we used another Python script to process all this raw data. Our script reads a data set from a CSV file and uses peak detection on the electrical current data to detect where measurements occurred. Then, it follows the graph down on the left and right side of the peak until it reaches a threshold value t that represents the idle electrical current draw of the microcontroller. The sample *left* of the peak is marked as that measurement's *beginning*, and the sample *right* of the peak is marked as that measurement's *ending*. The t-value functions as a baseline to keep these beginnings and endings consistent for individual measurements. This is necessary because the amount of electrical current the MCU draws highly fluctuates when the device

is in deep sleep. We found that the median of the measured current[11] serves as an accurate cut off value. Due to the fact that the device spends a relative large portion of its time in a deep sleep state, approximately 95% of the sampled data is in this fluctuation range. By using the median instead of the average, we ensure that the peak values we use for our measurement (which are only in the top 5% of our data) are not actually used to determine a proper cut-off value t.

4.3 Results

In this section we present our findings and how we came to them. First we explain how we applied a preselection to our data set in order to filter out redundant or irrelevant information. We then divide our data into two categories: *AEAD* and *non-AEAD* cipher-mode combinations and show which of the evaluated ciphers performs best in each category. AEAD often comes with a performance penalty and whether it is required depends mostly on what exactly is being protected. With this division between cipher-mode combinations that do and do not offer integrity protection, we can clearly see the exact performance penalty that is incurred by this additional security. The reader can decide for themselves whether that penalty is worth it given the increase in security.

4.3.1 Preselection

We removed two sets of samples from our data set. The first set contains all samples of the `Blowfish` cipher. As can be seen in Appendix A.3.1, `Blowfish` had the highest duration and CPU usage, and its average current is amongst the highest few, too. From a security perspective, the cipher is technically still good for use. However, although the cipher has not been broken yet, even its designer Bruce Schneier no longer recommends it in the current day and age[12]. Based on these findings, we decided to no longer consider `Blowfish` for use in our design.

The second set of samples we omitted are those with an `ECB` mode of operation. From our initial threat assessment we learned that attacks on *data confidentiality* and *data integrity* can cause the most damage to a DMS network. Therefore, a cipher or mode of operation that is fundamentally unable to protect against attacks on those principles should never be considered for use. Rogaway [32] shows that `ECB` has very weak security properties and does not achieve any generally desirable security goal. He states it should only ever be used as a building block for other modes of operation. As such, we decided to omit this mode of operation from our results entirely.

Figure 9 shows what our data set looks like after the preselection. Each point in the plot represents a unique cipher-mode-key combination, averaged over fifty measurements. One interesting observation that immediately stands out is that `AES` (blue), arguably the most used symmetric cipher, displays the highest values for all performance metrics. However, as soon as it is accelerated by hardware (`AES(HW)`, orange) it shows the best performance

[11] This excludes samples from chip initialization in the beginning and deinitialization at the end of benchmark runs.

[12] https://www.schneier.com/academic/blowfish/

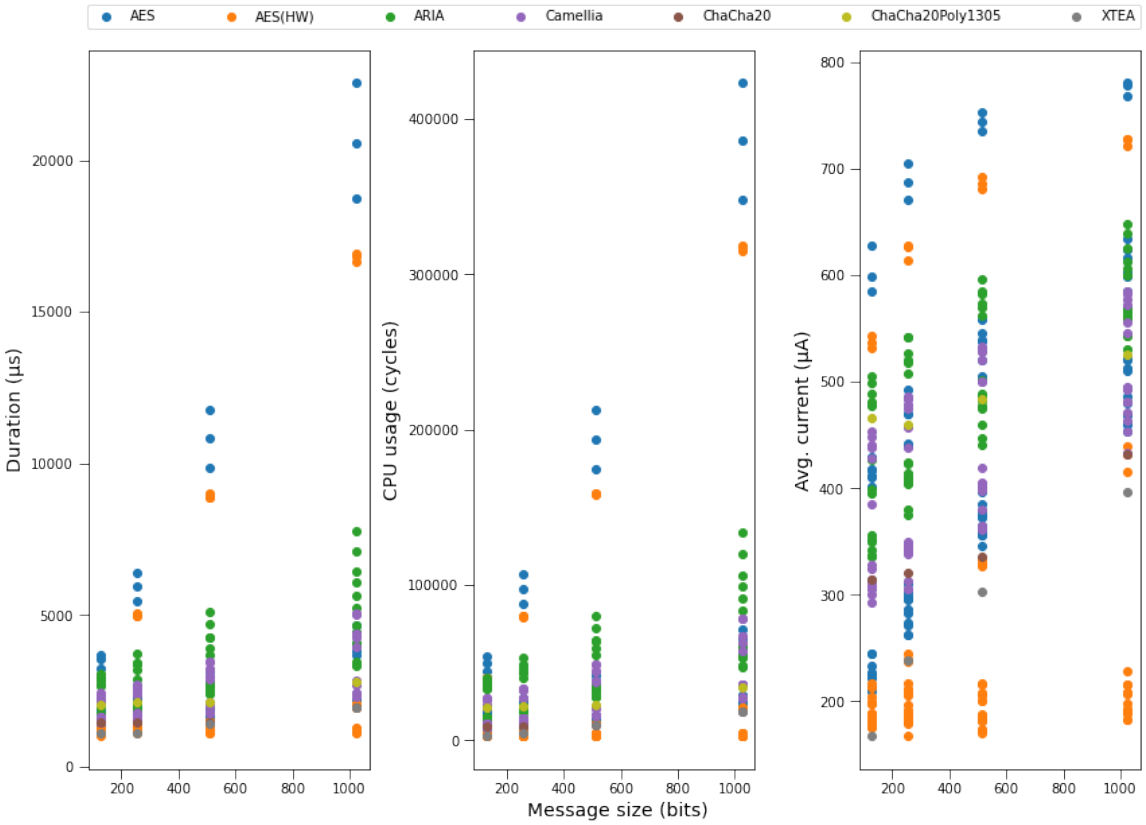


Figure 9: A scatter plot showing the spread of performance on all three performance metrics, after the preselection has been applied. Each point represents a unique cipher-mode-key combination. For all three performance metrics a higher value indicates a worse performance.

over all three metrics, beating ciphers specifically designed for small payloads in the process.

4.3.2 AEAD versus non-AEAD

We now split our data into two categories: encryption methods with and without support for AEAD. The only cipher in our list that inherently provides authenticated encryption is `ChaCha20-Poly1305`, with `ChaCha20` being its non-AEAD counterpart. The other ciphers only provide authenticated encryption if they are used in conjunction with either the `CCM` or `GCM` mode of operation. By categorizing the data in this way, we are able to visualize the performance penalty that using authenticated encryption incurs. Figure 10 shows how the energy consumption of a cipher increases as authenticated encryption is used. Note that Figure 10 is quite heavily zoomed in. The original figure showing all data can be found in Appendix A.3.2. For this analysis, we will focus on the bottom section of the graph, as shown in Figure 10. Here we clearly see that for `AES, ARIA, Camellia,` and `ChaCha20,` the usage of authenticated encryption leads to a significant increase in energy consumption. Only for `AES(HW)` this does not happen. We believe this is caused by the MCU's hardware being able to very efficiently process and move large chunks of data. If

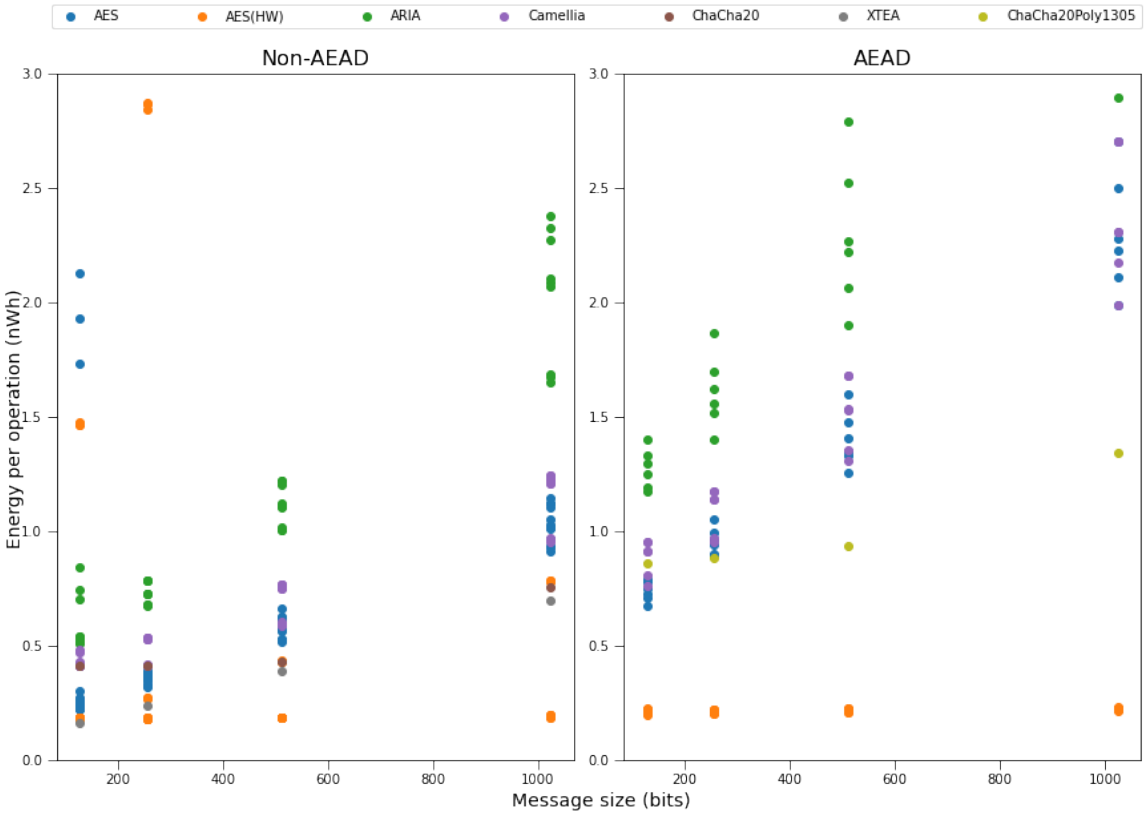


Figure 10: A zoomed in scatter plot showing the difference in energy consumption of AEAD and non-AEAD encryption. For most ciphers (distinguishable by the different colors), the average energy consumption increases when authenticated encryption is used. Only `AES(HW)` remains nearly constant as message size increases. The `AES(HW)` outlier in the non-AEAD graph is caused by the `CFB8` mode of operation, which was the only mode to not exhibit constant behavior.

we were to determine an imaginary "winning cipher" for both categories, i.e. the cipher that shows the lowest overall energy consumption, `AES(HW)` wins in both cases. In terms of energy consumption per encryption operation, it is surpassed only once: `XTEA-128-CBC` shows the absolute lowest energy consumption measured at ~0.162 nWh per encryption. However, even then `AES(HW)-192-CBC` is a close second with a consumption of ~0.177 nWh, while even providing the increased security of a 192-bit key.

The final step in the analysis of our results is determining which cipher-mode combination can be considered "the winner". To do so, we first briefly disregard key and message sizes by grouping and averaging our data set. As we show in Appendix A.3.3, every cipher scores worse in all performance metrics as the size of the used key or message increases. The only exceptions to this rule are `AES(HW)` and `Camellia`, whose performance remains roughly constant as the key size increases. However, we have already shown that `AES(HW)` performs better than the other ciphers and `Camellia` only displays constant behavior in the jump from a 192-bit key to a 256-bit key. Therefore, we believe that no important information is lost by temporarily grouping and averaging data. Furthermore, in doing so,

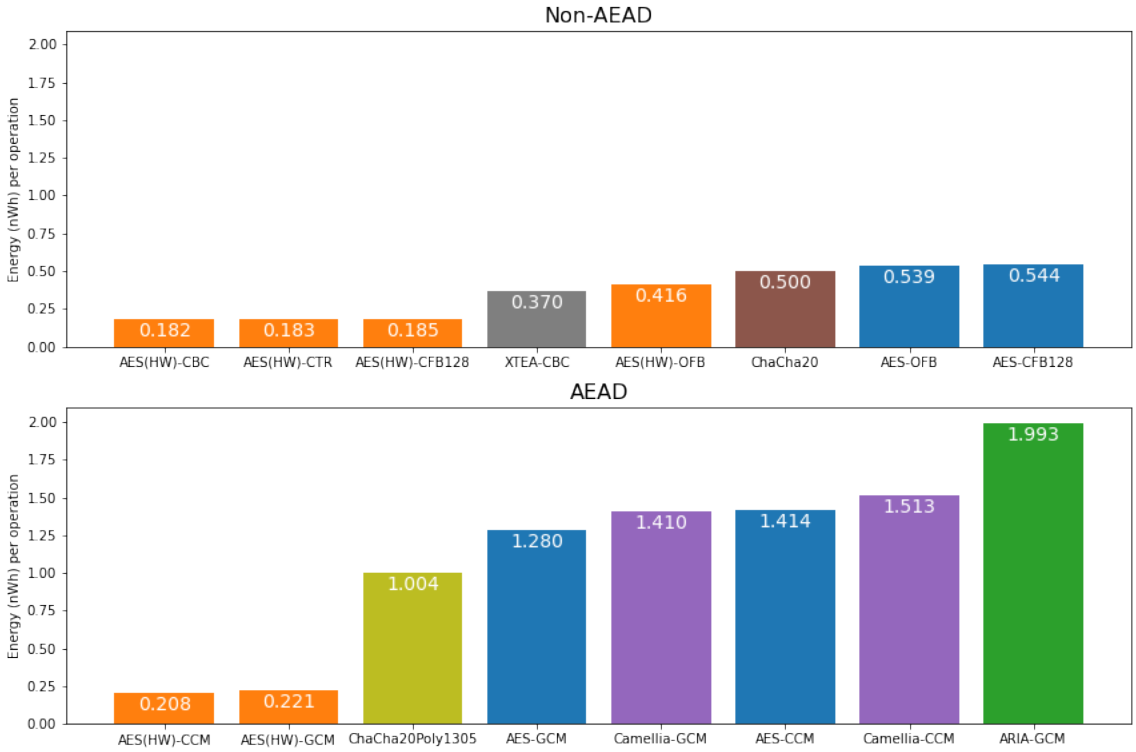


Figure 11: These bar charts show the eight best performing cipher-mode combinations in the AEAD and non-AEAD categories, in terms of energy consumption per encryption operation.

we are able to compare different cipher-mode combinations as a whole, without having to consider different performance levels for different keys and messages.

As displayed in Figure 11, `AES(HW)` consumes the least energy in both categories. In the non-AEAD category, the `CBC` mode, closely followed by `CTR` and `CFB128`, consumes 50.2% less energy than the closest other cipher, `XTEA`. In the AEAD category, both `CCM` and `GCM` lead by a large margin, with a decrease in energy consumption of 79.3% when compared to the next closest cipher, `ChaCha20-Poly1305`. Both `CCM` and `GCM` internally use `CTR` for their actual encryption, which is one of the most efficient non-AEAD modes. Nonetheless, the interesting aspect here is that hardware accelerated encryption outperforms software encryption in almost all cases. Even when the software encryption has been specifically designed for use in IoT networks, where the payloads are often small and computing resources scarce, a hardware implementation of a traditionally "less efficient" cipher still performs better in all aspects.

4.4 Conclusion

Now that we have seen which cipher-mode combinations perform the best, we can apply this information to what we know about the DMS and determine which method of encryption we will be using in our final design. To decide which of the evaluated cipher configurations performs best for the DMS, we first need to take a closer look at what exactly happens

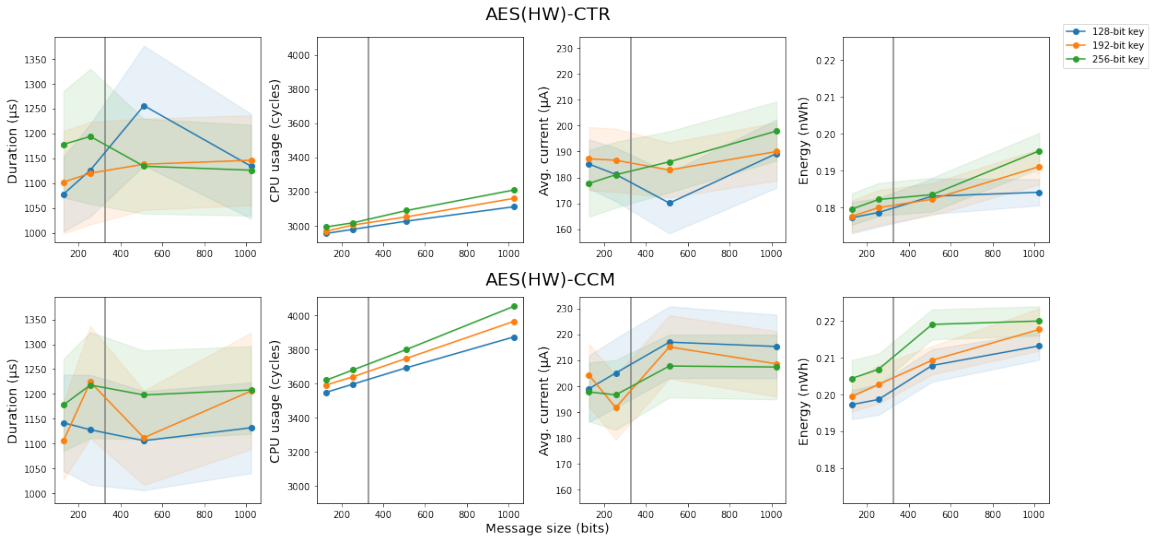


Figure 12: A comparison between `AES(HW)-CTR` and `AES(HW)-CCM` for all three performance metrics and the total energy consumption, including 95%-confidence intervals. The most occurring message size according to 's protocol (328 bits) has been indicated with a grey line.

within the network. Most of the messages sent by the Animal Trackers are relatively short in length. The most frequently sent message has a length of 41 bytes (or 328 bits), with larger messages of up to 100 bytes (800 bits) taking the second place.

Although `CBC` actually performed the best, according to our results, we ultimately decided to use `CTR` instead. `CBC` mode has been proven to come with quite a few weaknesses. Bellovin [33] has demonstrated a number of practical real-world attacks on the mode of operation. Additionally, `CTR` has the added benefit that it transforms any block cipher into a stream cipher. This removes the need for padding, which likely outweighs the marginal increase in energy consumption as shown in Figure 11.

Figure 12 presents an overview of how `AES(HW)-CTR` and `AES(HW)-CCM` perform in the performance metrics. The grey line indicates a message size of 328 bits, which is the most occurring message size. These graphs confirm what we previously claimed about how performance decreases as key size or message size increases. In terms of an encryption's *duration*, our results show that there is a lot of fluctuation in the data. This is very likely a result of our method for determining when a single encryption operation begins and ends, as was explained in Section 4.2.3. However, the practical consequence of a 1000 μs versus a 1350 μs encryption is negligible. When looking at *CPU usage*, *average current,* and *energy consumption*, we see a slightly larger difference between the AEAD and non-AEAD solutions.

To gain a little more insight into how both options perform over ten years (the expected lifetime of a single Animal Tracker), we computed the power draw over that period of time. Table 10 contains the results of these computations. In previous figures, we used the physical quantity *energy* (nWh) to denote the energy consumption of an encryption

Mode	$\mathbf{I}(\mu\mathbf{A})$	$\mathbf{t_{enc}}(\mu\mathbf{s})$	$\mathbf{t_{1h}}(\mathbf{s})$	$\mathbf{t_{10y}}(\mathbf{h})$	$\mathbf{C_{10y}}(\mathbf{mAh})$
CTR-128	175.648	1191	0.893	21.754	3.821
CTR-192	184.768	1129	0.847	20.622	3.810
CTR-256	183.548	1164	0.873	21.261	3.902
CCM-128	211.031	1117	0.838	20.403	4.306
CCM-192	203.425	1168	0.876	21.334	4.340
CCM-256	202.207	1208	0.906	22.065	4.462

Table 10: An overview of how the total energy consumption of different encryption algorithms over ten years was computed. In this table, I is the measured average electrical current, and t_x and C_x are the time spent encrypting and consumed electric charge over x amount of time, respectively.

operation. For this table we have chosen to instead use *electric charge* (mAh), as this is more in line with standards in the embedded hardware field. We first estimated how many messages are sent by an Animal Tracker per hour by dividing the total number of bytes sent per hour by the most occurring message size (328 bits). This resulted in a total of approximately 751 messages. We utilized linear interpolation to estimate the encryption duration and average electrical current seen in the table. In Figure 12 these values were indicated with the vertical grey line. Finally, with these estimated values, we computed how much electric charge the encryption operations use over a period of ten years.

Table 10 shows that even AES(HW)-CCM with a key size of 256 bits uses only approximately 4.5 mAh over a period of ten years. Earlier, we stated that an Animal Tracker has a nominal capacity of 2,000 mAh, meaning that the addition of AES(HW)-256-CCM encryption would use roughly 0.2% of the Animal Tracker's total energy capacity. With a shorter key size or by using the CTR mode of operation, this energy consumption can be reduced even further. However, considering the additional security benefits that CCM offers over just CTR, we decided to use CCM in our final design.

5 Key Agreement Evaluation

In the third section of this research, we finish our system design by implementing and evaluating an authenticated key agreement protocol suitable for use in an IoT environment. Like in our previous symmetric cipher evaluation, we used a Silicon Labs EFR32MG22 MCU for the evaluation. The Cortex-M33 core of the MCU was again running at a clock speed of 20 MHz with FSRCO as the clock generation source. The purpose of this evaluation was not necessarily to compare different methods or approaches, but rather to assess whether such a key agreement scheme is feasible for use in IoT. First, we provide the theoretical background information necessary to understand this section. Second, we describe and discuss the protocol we have implemented and its security benefits and shortcomings. Finally, we present the evaluation results and decide whether the scheme is feasible for practical use.

5.1 Background

5.1.1 Key Agreement

A key agreement protocol allows two or more parties without a secure communication channel to jointly construct a session key that they can then use for symmetric encryption. One of the first examples of a practical key agreement protocol is that of Whitfield Diffie and Martin Hellman. This protocol is better known as the *Diffie-Hellman (DH) key exchange* [34]. It was created in 1976 and still forms the basis for many key agreement protocols today. The latest version of the *Transport Layer Security* (TLS) protocol (version 1.3) still uses a DH-based key exchange. There exist two major variants of Diffie-Hellmann: a static and an ephemeral version. In both versions the parties use their long-term secret key (private key) to construct a session key. In the static version, parties always use the same secret information when constructing a session key. Ephemeral Diffie-Hellmann (EDH), conversely, randomizes parts of the secret information. This enables *forward secrecy*, which means that a compromised long-term secret key of one (or more) of the parties does not compromise session keys of terminated sessions. In other words, if an adversary passively captures encrypted traffic in the hope of one day compromising both party's secret keys, he will still fail to decrypt the captured data. The session key that was used to encrypt the traffic he captured cannot be reconstructed from just the long-term secret keys, but also requires knowledge of the ephemeral secret information.

5.1.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography, often abbreviated as ECC, is a form of public-key cryptography based on elliptic curves. The use of such a mathematical construction for cryptographic purposes was suggested by Miller [35] and Koblitz [36] in 1985. It took over three decades for ECC to become widely adopted. However, since then it has been recommended numerous times by official organizations (NIST [37], SECG [38, 39, 40]) and standards (TLS [41]). Other large projects, such as the cryptocurrencies Bitcoin [42] and Ethereum, also use elliptic curve cryptography. Compared to non-EC cryptography, ECC allows for smaller keys while providing equivalent security [43]. In terms of efficiency, a 160-bit ECC private

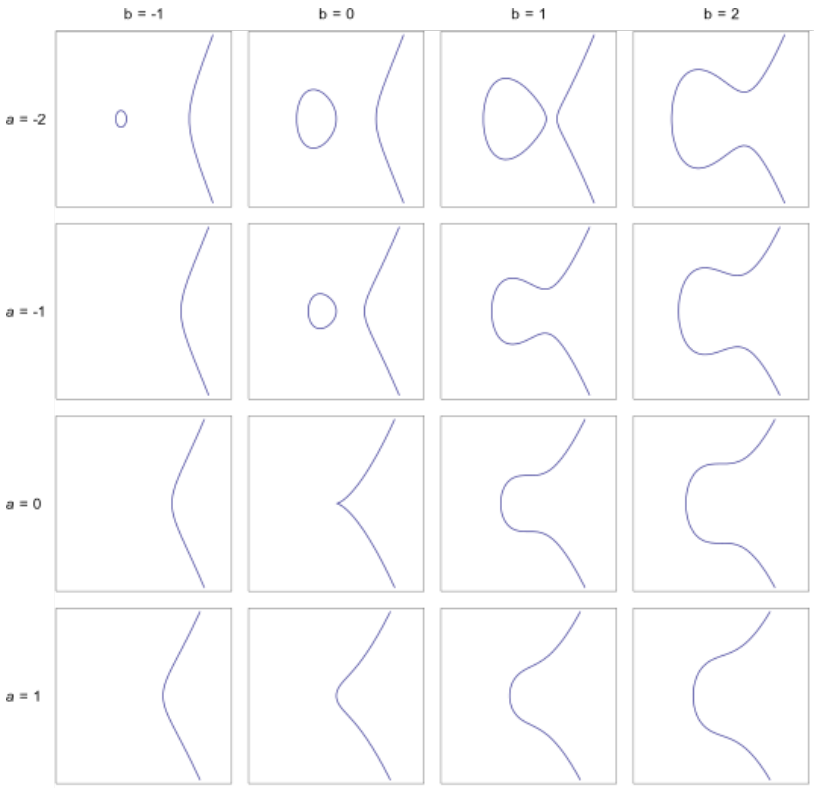


Figure 13: An overview of elliptic curves in the range with parameters $a \in [-2, .., 1]$ and $b \in [-1, .., 2]$. The curve with $(a, b) = (0, 0)$ has a singular point in the form of a cusp and can therefore not be an elliptic curve. Image source: [44].

key is roughly equal to a 1024-bit RSA key. Using a smaller key requires less resources during cryptographic operations, often making ECC faster and more efficient than non-EC cryptography.

In cryptography, an elliptic curve is a plane curve over a finite field[13] (also called Galois field or GF) containing the points that satisfy the equation

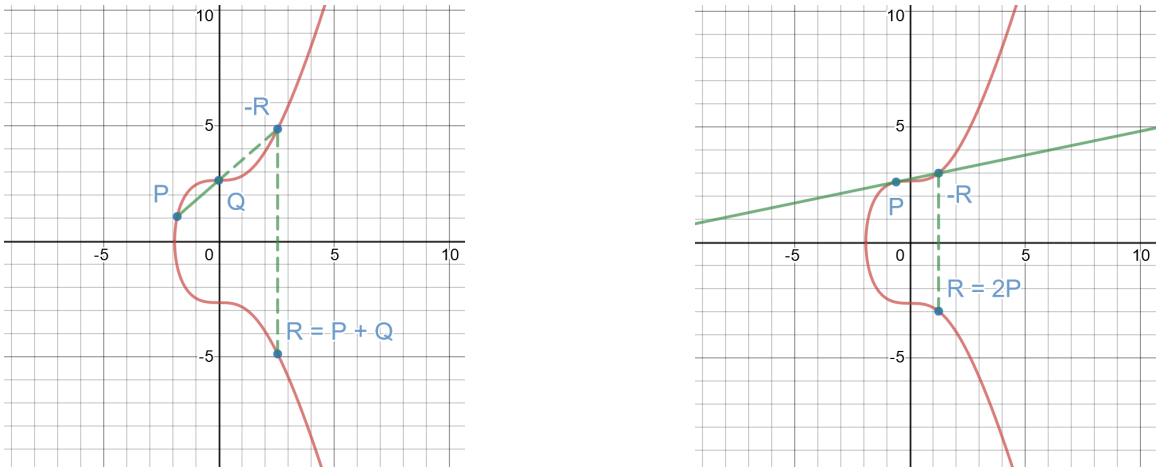
$$y^2 = x^2 + ax + b \tag{4}$$

and a distinguished point at infinity (∞). Elliptic curves defined by this equation are called Weierstrass curves. Other types of curves exist, with different advantages and disadvantages. However, these will not be discussed in this work. The definition of an elliptic curve also states that it must be *non-singular*, meaning it cannot have cusps, self-intersections or isolated points. Figure 13 shows sixteen curves (of which fifteen are elliptic curves) for different a and b parameters. Note that the curve with $(a, b) = (0, 0)$ has a cusp (or singular point), which means it is not an elliptic curve. Elliptic curves are defined by a set of domain parameters. These domain parameters define the properties of the cryptosystem that all parties agree on. Generally these public domain parameters include the following:

[13] A finite field is a set of finite length that satisfies a list of special properties, called *field axioms*. A common example of a finite field is the set of all integers modulo p where p is a prime number.

- p: the prime of the finite field $\mathbb{F}_p$,
- a: the first coefficient of the elliptic curve,
- b: the second coefficient of the elliptic curve,
- G: the cyclic subgroup of $\mathbb{F}_p$ containing all points on the elliptic curve: $G : \{(x, y) : x, y \in \mathbb{F}_p\} \cup \{\infty\}$,
- P: the generator (or base point) of G,
- n: the size (cardinality) of G.

Additionally, there are three basic operations defined on elliptic curves: *point addition*, *point doubling* and *scalar multiplication*.



(a) Point addition **(b)** Point doubling

Figure 14: Point addition and point doubling shown on the `secp256k1` elliptic curve ($y^2 = x^3 + 7$) as defined by SECG [39]. It should be noted that these geometric representations are of the curves over real values and not over $\mathbb{F}_p$. This is done for visual clarity only, as the operations remain the same for both types.

Point addition is the addition of two points P and Q on an elliptic curve, to obtain another point $R = P + Q$ on the curve. Figure 14a shows a geometric representation of point addition. To add the two points P and Q, a line is drawn through both points. This line intersects the elliptic curve at exactly one more point, point $-R$. This point $-R$ is then mirrored with respect to the x-axis to arrive at point R, the result of $P + Q$. The coordinates of point R can also be computed algebraically. First, the slope s of the line running through points P and Q is computed with $s = \frac{y_P - y_Q}{x_P - x_Q}$. The coordinates of R are then calculated with $x_R = s^2 - x_P - x_Q$ and $y_R = -y_R + s(x_P - x_R)$. Note that when the

points P and Q are each other's mirror points (with respect to the x-axis), the resulting point R is the point at infinity.

Point doubling is the addition of a point P on an elliptic curve to itself, to obtain another point $R = 2P$ on the curve. Figure 14b shows a geometric representation of point doubling. To double a point P, first the line tangent to the elliptic curve in point P is drawn. This line intersects the elliptic curve at exactly one more point, point $-R$. This point $-R$ is then mirrored with respect to the x-axis to arrive at point $R = 2P$. Like point addition, point doubling can also be performed in an algebraic manner. The formulas to compute the coordinates (x_R, y_R) are identical to those the point addition operation, but the slope line s is now calculated as $s = \frac{3x_P^2+a}{2y_P}$, where a is from the elliptic curve equation.

Scalar multiplication is the multiplication of a point P on an elliptic curve with a positive integer k, to obtain another point $R = kP$ on the curve. The naive method of performing scalar multiplication is simply performing a series of k point additions, such that $R = \sum_1^k P$. However, this computation rapidly becomes very costly as in practice, k is often a very large integer. When using a 256-bit elliptic curve, k can be as large as $2^{256} - 1$, which would make a single scalar multiplication as efficient as a brute force attack. Fortunately, alternative approaches have been developed to speed up and optimize scalar multiplication. Some commonly used scalar multiplication techniques [45] are *double-and-add*, using the *NAF* (Non-Adjacent Form)/*width w-NAF* algorithm, a *sliding window* approach, or using the *Montgomery ladder* algorithm. We will not discuss these techniques in more detail. Ramdani et al. [46] have evaluated the performance of a number of different scalar multiplication techniques and have found that using the NAF algorithm results in the lowest energy consumption. They also show that using a Montgomery ladder leads to a fixed execution time for scalar multiplication. This is helpful against timing-based side-channel attacks. The strength of elliptic cryptography is based on the discrete logarithm problem (DLP). This problem states that given points $P, Q \in G$ (where Q is a multiple of P) it is difficult to find an integer k such that $Q = kP$. The most efficient method of finding such an integer is by trying all possible integers in $\mathbb{F}_p$. This has a computational complexity of $\mathbb{O}(n)$, which quickly becomes very impractical for a large k.

5.1.3 Certificate-based Cryptography

A major disadvantage of public-key cryptography is that it is difficult to authenticate public keys. In theory, anyone is able to broadcast a public key and pretend to be someone else. Without authentication, it is impossible to distinguish a "fake" public key from a legitimate one. Consider the following example. Alice and Bob want to establish a secure channel to communicate over. They decide to use ECDH, a Diffie-Hellmann key exchange based on elliptic curves. Figure 15 shows how this protocol works. They generate their secret keys d_A and d_B and multiply it with the group generator P to obtain their public key Q. Alice and Bob then send this public key to the other each other over an insecure channel. Unbeknown to both, Mallory, a malicious adversary, intercepts these messages. She generates two key pairs, one for Alice and one for Bob. She sends her public keys to their respective receivers and from then on is able to decrypt messages sent by both Alice and Bob. To ensure that

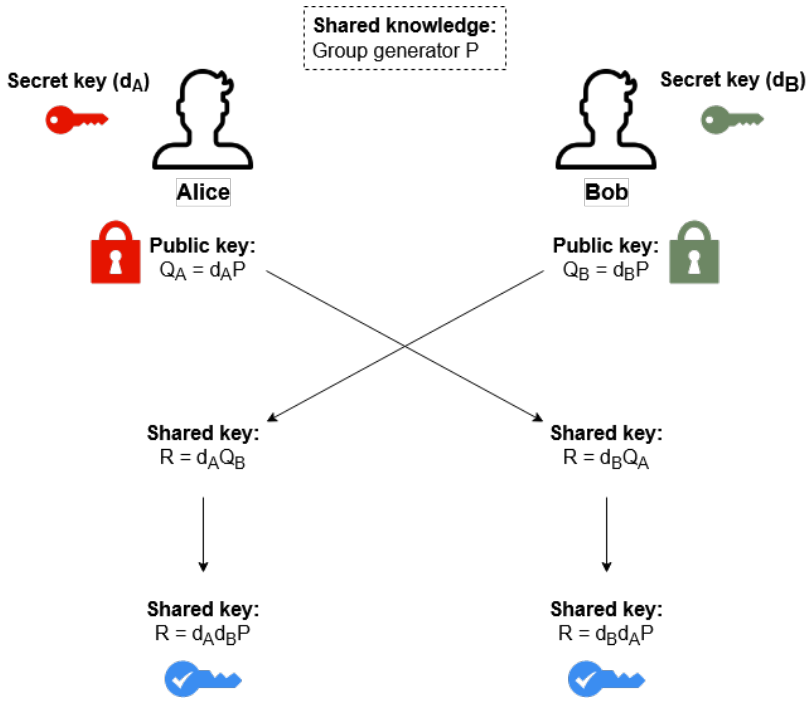


Figure 15: A diagram showing how Elliptic Curve Diffie-Hellmann (ECDH) works. Without external authentication, this protocol is vulnerable to a man-in-the-middle attack. An active adversary can intercept the Q_A and Q_B messages and pretend to be Alice to Bob, and Bob to Alice.

Alice and Bob do not find out that their secure channel is compromised, Mallory simply re-encrypts the decrypted messages and forwards them to the intended receiver.

It is clear that authentication is important when using a public-key encryption scheme. In practice, a public-key infrastructure (PKI) is often used. A PKI is a set of roles, policies, and procedures to manage a public-key cryptosystem. It uses a *certificate authority* (CA) to guarantee public key authenticity. Every node in the network receives a certificate from the CA with a unique signature that only the CA could have created. This allows other parties to verify that the node holding that certificate (and its public key) has been validated by the CA. The downside to a certificate-based approach is that it causes a lot of overhead. The CA is often required to remain online in the network, which requires additional infrastructure. Furthermore, the certificates themselves can grow quite large in size. To illustrate this, we generated a 3072-bit RSA key pair and a 256-bit ECC key pair. Both of these key pairs provide approximately 128 bits of security [47]. We then generated a certificate for both key pairs. The RSA certificate had a size of 1,135 bytes and the ECC certificate had a size of 728 bytes. For powerful devices such as computers and smartphones, such sizes are not an issue. For constrained IoT devices, however, transmitting these certificates may require multiple messages, depending on how many bytes the device can send in a single message. This shows that even an ECC certificate, known for its smaller size, might still be too large for practical use in IoT.

Step	Point add.	Scalar mul.	Hashes	Random no. gen.
Setup	0	1	0	1
Extract	0	1	1	1
Verify	1	2	1	0
Key Agreement	2	5	2	1

Table 11: An overview of the PF-IDAKA protocol showing each protocol step broken down into its primitive components.

5.1.4 Identity-based Cryptography

In identity-based cryptography, a node's public key is derived from its public identity. This means that any party with access to the network's public domain parameters is able to compute a node's public key by using that node's identity. The corresponding private key is generated by a party, *Key Generation Center* (KGC), that is trusted by all network nodes. This is done to ensure that only legitimate nodes receive a private key that matches their identity. Identity-based cryptography removes the requirement for certificates, because only a valid identity can be converted into a public key that matches a private key. In other words, a "fake" identity leads to a public key for which there is no corresponding private key, making decryption of messages impossible.

Identity-based (ID) cryptography was first proposed by Shamir in 1984 [48], in the form of an ID-based signature scheme. Actual ID-based encryption remained an open problem until Boneh and Franklin [49] proposed a scheme in 2001. Since then, ID-based cryptography has been heavily researched and different types of schemes have been proposed, such as key agreement protocols [50, 51] and specialized authentication schemes [52]. Up until recently, many schemes were built on mathematical primitives that are not suitable for use in IoT, such as bilinear pairings. A bilinear pairing is a mathematical tool that maps two elements in one elliptic curve group to an element in a related finite field. Pairings are costly operations and schemes that utilize them are generally not practical for use in a constrained setting. Cao et al. [50] evaluated the performance of the pairing operation on a non-constrained platform. They found that a single ECC scalar multiplication takes 0.83 milliseconds, whereas a single pairing operation takes 20.01 milliseconds; that is an increase of over 23,000%. A similar increase in execution time is to be expected on a constrained device.

5.2 PF-IDAKA Protocol

We have implemented a pairing-free identity-based authenticated key agreement scheme (PF-IDAKA) with minimal message exchange by Cao et al. [50]. This scheme allows an Animal Tracker and a PU to securely and efficiently compute a long-term symmetric encryption key. Unlike many other ID-based key agreement schemes, it does not use bilinear pairings, as it relies solely on ECC operations. Furthermore, it exchanges only two messages between a PU and an Animal Tracker. This is a highly desirable feature, as the connection between a PU and an Animal Tracker is not always guaranteed to be stable.

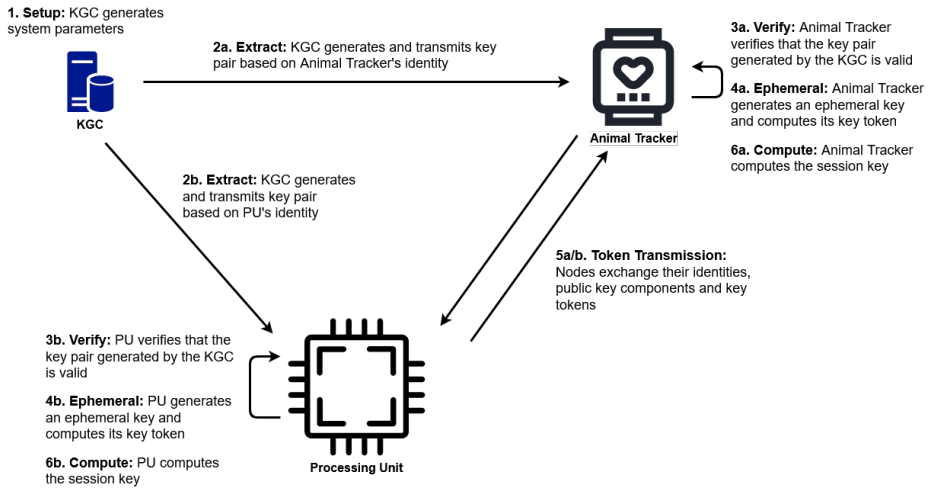


Figure 16: A high-level overview of the PF-IDAKA protocol. Steps 1 and 2 (and optionally 3) are performed during the manufacturing of the PU and the Animal Tracker. Steps 4-6 are performed when the user wishes to add a new Animal Tracker to his system.

The protocol consists of four steps: *Setup*, *Extract*, *Verify* (optional) and *Key Agreement*. Furthermore, there are three parties involved: a KGC and two different nodes. Figure 16 shows a high-level overview of the protocol. Note that in the figure, the *Key Agreement* step has been split into three substeps: *Ephemeral*, *Token Transmission* and *Compute*. This division is only used in this graphical representation to better illustrate how the protocol works. In the written protocol description that follows, we instead follow the steps as indicated in the original research paper.

The KGC in the context of this DMS is located at the facility where Animal Trackers are produced. The two nodes in this context are a PU and an Animal Tracker. In the *Setup* step, the KGC is initialized and the system's domain parameters are determined. This includes all parameters listed in Section 5.1.2, as well as the system's public key and two hash functions. The KGC also generates its master private key. The next step, *Extract*, is performed continuously as new Animal Trackers are produced. In this step, the KGC generates a unique key pair using an Animal Trackers's identity and the system's domain parameters. Because of the properties of ID-based cryptography, the Animal Tracker's identity and key pair are mathematically related. This property allows a PU to later verify that the node he wishes to communicate with is authenticated. The *Verify* step is optional and allows a node to verify whether their key pair was generated correctly. This is useful in case the nodes in a network do not necessarily trust the KGC. In this case, though, this step is not required, as the entire system is designed and built by a single party. The final *Key Agreement* step occurs when the Animal Tracker is attached to a live DMS network. The PU and the Animal Tracker both generate an ephemeral key and from that, a key token. They then exchange these key tokens, after which they can both derive the same session key. This session key is from then on used as the long-term symmetric encryption key. Table 11 shows how these four steps are broken down into the primitive operations of point addition, scalar multiplication, hash computation, and random number generation.

Based on this breakdown, *Key Agreement* is likely to be the most expensive step in the protocol.

In terms of message size, the PF-IDAKA protocol benefits from the use of elliptic curve cryptography. The messages that the two parties transmit to each other contain their identity and two points on the curve. If compressed curve point notation is used, this comes down to a total size of $b_{\text{ID}} + 2 \cdot (1 + b_n)$, where b_{ID} and b_n are the byte size of an identity and the byte size of a single point on the curve, respectively (n is the order of the group G that is formed by all curve points).

5.2.1 Security

Cao et al. prove the security of their protocol using Kudla and Paterson's modular approach [53] based on the Modified Bellare-Rogaway model (mBR model) [54]. The protocol provides strong security features, including key compromise impersonation resilience, perfect forward secrecy and master key forward secrecy. However, Islam et al. [51] have found that the scheme is vulnerable to at least two attacks, namely a *key offset attack* (KOA) and a *Known session-specific temporary information attack* (KSTIA). In the KOA, the adversary offsets the session key by an exponent σ without this being known to the PU and the Animal Tracker. Although this attack does not provide the adversary with any information about the session key, it does violate the integrity of the key, which can be considered a security risk under certain circumstances. However, considering the PU and the Animal Tracker communicate over a local network, the adversary would have to be physically near the farm to perform this attack. Recall from our threat assessment in Section 3.2.1 that the only attacker capable of performing such a technical attack is the Cyber Criminal. However, as we identified, it is very unlikely for that attacker to have the physical network access required to perform this attack. As such, we estimate the threat of a KOA to be negligible. The KSTIA relies on insecure storage of random session-specific parameters (the ephemeral key). In this context, a KSTIA is not an additional threat, since an attacker who can extract information from insecure storage can also extract the session key after it has been established. To summarize, although the scheme we have implemented has its weaknesses, we do not believe these weaknesses to be relevant for our use case.

5.3 Methodology

For the evaluation of our key agreement protocol we used roughly the same tools and methods as we did for the symmetric cipher evaluation (see Section 4.2). Recall that the purpose of this evaluation is only to assess the practical feasibility of such a key agreement scheme for IoT networks. As such, this evaluation is less extensive than our previous one. We first briefly mention the tools we have used and then describe our methodology.

Tooling *Simplicity Studio 5* was used to write, build, and upload code to the MCU. Energy consumption measurements were performed using Simplicity Studio's Energy Profiler, similar to the last time. We also reused *Mbed TLS* for its elliptic curve cryptography implementation. The Silicon Labs distribution of this library contains hardware acceleration for

certain elliptic curve operations, which allowed us to measure the change in performance caused by using specialized hardware.

5.3.1 Design

A practical implementation of the chosen PF-IDAKA scheme requires an elliptic curve of k bits (the security parameter). We have chosen $k = 256$, which provides approximately 128 bits of security. Mbed TLS supports a number of different elliptic curves that provide this level of security, but only a few have a Silicon Labs hardware accelerated implementation. The curve we have chosen is the NIST *secp256r1* curve. Due to limitations of Mbed TLS, we were not able to use an elliptic curve with 256 bits of security for our implementation, which would match the bit strength of `AES-256`. When using a key exchange scheme with n bits of security and an encryption scheme with m (where $n < m$) bits of security, the overall system only has n bits of security [55]. We therefore strongly urge readers to at least use a 512-bit elliptic curve in practice. This *secp256r1* curve has the following domain parameters:

- $p =$ `FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF`

  $p = 2^{224}(2^{32} - 1) + 2^{192} + 2^{96} - 1$
- $a =$ `FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC`

  $a = -3$ (this is a special value[14])
- $b =$ `5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B`
- $P =$ `03 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C96`

  (in compressed form[15])
- $n =$ `FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551`

We evaluated the performance of the PF-IDAKA scheme implementation using the following three performance metrics. Note that these metrics are equal to those used during our symmetric cipher evaluation.

1. The duration of a single operation, measured in microseconds (μs),
2. The CPU usage of a single operation, measured in CPU cycles,
3. The average electrical current a single operation draws, measured in microamperes (μA).

[14] An a value of -3 allows for more efficient point doubling due the way in which this operation is defined.

[15] In compressed form, indicated by starting byte `0x03`, only the x-coordinate of a point is given. The y-coordinate can be derived by solving the elliptic curve equation for the given x-coordinate.

Step	HW	Duration (μs)	Avg. current (μA)	CPU load (cycles)
Setup	✓	$18\,220 \pm 36$	907.41 ± 1.74	$346\,330 \pm 117$
	✗	$4\,124\,160 \pm 2729$	745.58 ± 0.29	$81\,151\,892 \pm 123\,058$
Extract	✓	$25\,855 \pm 106$	847.11 ± 2.20	$488\,613 \pm 3487$
	✗	$4\,133\,495 \pm 4218$	744.51 ± 0.18	$81\,472\,868 \pm 118\,262$
Verify	✓	$44\,270 \pm 76$	887.50 ± 1.02	$852\,025 \pm 3560$
	✗	$8\,426\,475 \pm 4882$	745.15 ± 0.09	$166\,354\,904 \pm 331\,374$
Key Agreement	✓	$99\,929 \pm 419$	896.99 ± 5.07	$1\,582\,243 \pm 3577$
	✗	$21\,095\,442 \pm 24\,982$	747.08 ± 0.14	$334\,821\,820 \pm 565\,855$

Table 12: The results of our key agreement protocol evaluation. The shown measurements are averaged over ten evaluation runs to eliminate measurement errors. The HW column indicates whether hardware acceleration was turned on or off during that evaluation run.

In practice, the Setup and Extract steps are performed by the KGC, which very likely is a far more powerful device than the MCU that we have built our implementation for. Nevertheless, for the sake of completeness, we have also evaluated how these steps perform on a constrained device, but this is not part of our system design.

5.3.2 Execution

To measure our three performance metrics, we used the same techniques as we did during our symmetric cipher evaluation. The CPU usage was measured by resetting and then reading the `CYCCNT` register on the MCU, whereas the average electrical current and operation duration were measured with the Energy Profiler in Simplicity Studio. We again put the MCU in a deep sleep state for 200 milliseconds *before* and for 100 milliseconds *after* every operation. This generated energy measurements that are similar to those we generated previously (as seen in Figure 8), which allowed us to reuse the Python analysis program we had developed previously.

Although the purpose of this evaluation is not to gather perfectly accurate results, it is still important to rule out any inconsistencies caused by measurement errors or other random events. Therefore, we evaluated all protocol steps exactly ten times for both the accelerated and non-accelerated variants of our implementation.

5.4 Results

Table 12 contains the results of our key agreement protocol evaluation. These results confirm what we briefly mentioned earlier: the *Key Agreement* step is by far the most expensive step in the protocol. It requires more time than all other steps combined in both the accelerated and non-accelerated variants. This is likely due to the high number of scalar multiplications in this step. The second most expensive step is the optional key verification. From these numbers we can clearly see the fact that enabling hardware acceleration for elliptic curve cryptography greatly increases the efficiency and speed of the operations involved. The most expensive step in the protocol, the actual *Key Agreement* as performed by an Animal Tracker, shows a duration decrease from approximately 21 seconds to less than a hundredth of a second; that is a decrease of 99.5%. In terms of CPU load, the usage drops from over 330 million cycles to just 1.5 million cycles; that too is a decrease of 99.5%. Only

Actor	Step	HW	I(μA)	t(μs)	C(mAh)
KGC	Setup	✓	907.41	18220	$4.59 \cdot 10^{-6}$
		✗	745.58	4124160	$8.54 \cdot 10^{-4}$
	Extract	✓	847.11	25855	$6.08 \cdot 10^{-6}$
		✗	744.51	4133495	$8.55 \cdot 10^{-4}$
Node	Verify	✓	887.50	44270	$1.09 \cdot 10^{-5}$
		✗	745.15	8426475	$1.74 \cdot 10^{-3}$
	Key Agreement	✓	896.99	99929	$2.49 \cdot 10^{-5}$
		✗	747.08	21095442	$4.38 \cdot 10^{-3}$

Table 13: The computed energy consumption of the different steps of the PF-IDAKA protocol. In this table, I is the measured average electrical current, and t and C are the duration and consumed electric charge of an operation, respectively. The HW column indicates whether hardware acceleration was turned on or off during that evaluation run.

the average electrical current during the operation increases when hardware acceleration is used. This is very likely caused by the fact that additional hardware is running on the MCU. When a software-based implementation is used, this hardware is disabled and thus the board draws less power overall. However, in terms of energy consumption, this increase in average electrical current is offset by the decrease in duration, as both are factors in computing the total energy consumption.

Another interesting observation is that the use of hardware acceleration seems to reduce the variability in the CPU load, while increasing variability in the average electrical current that is drawn. We believe the former is the result of the hardware drawing more power during certain primitive operations, whereas the CPU draws a stable current at all times. The latter phenomenon is likely due to the fact that the CPU simply has less work to do when hardware acceleration is used, therefore leaving less room for variability over multiple runs.

5.5 Conclusion

To determine whether the results shown in Table 12 are feasible for practical use, we once again computed the electrical discharge. Whereas we used ten-year timespan for the symmetric cipher evaluation, we only look at a single protocol run in this case. The reason for this is that key agreement is only performed once in most cases. The outcomes of our total energy computations are shown in Table 13. These results indicate that this ECC-based key agreement protocol is very feasible for practical use, both with and without hardware acceleration. With hardware acceleration, the total energy consumption per run of the protocol (including verification of the key) is $1.09 \cdot 10^{-5} + 2.49 \cdot 10^{-5} = 3.58 \cdot 10^{-5}$ mAh. Without hardware acceleration, this increases to $1.74 \cdot 10^{-3} + 4.38 \cdot 10^{-3} = 6.12 \cdot 10^{-3}$ mAh. Clearly, both a hardware accelerated and a non-hardware accelerated implementation can be used in practice. Their total energy consumptions represent only $1.79 \cdot 10^{-6}$% and $3.06 \cdot 10^{-4}$% of an Animal Tracker's total energy budget, respectively. The total duration, on the other hand, shows a decisive winner. Without hardware acceleration, a PU and an Animal Tracker would have to remain within close distance of each other for approximately thirty seconds

to successfully compute a joint session key. This duration is reduced to approximately 0.15 seconds when hardware acceleration is used. Although mechanisms for retrying key agreement can be implemented, a PU and Animal Tracker being able to perform a handshake within a second is definitely more convenient and less prone to errors.

In terms of message size, the PF-IDAKA protocol is very practical, too. With the parameters we used for our evaluation, the total size of a single message was only 81 bytes. This message contained a node's identity (15 bytes) and two elliptic curve points (33 bytes each). Compared to the 728 bytes ECC certificate we generated earlier, this is an improvement of 88.9%, and this is just considering the actual certificate that needs to be transmitted. Depending on the exact protocol, more messages could even be required. All in all, the PF-IDAKA protocol seems very fit for practical use in this context.

6 Conclusion

In this research, we designed and implemented a cryptographic system to secure data in a DMS, containing a large quantity of constrained devices. The main component of a DMS network are the Animal Trackers, which transmit sensor data to a PU. Animal Trackers are expected to operate on a very limited energy budget for multiple years. Therefore, next to providing adequate security, the main requirement for our system design was a low energy consumption. To achieve this, we first performed a risk analysis to determine which threats require the most attention. We developed a model to quantify the profiles of potential attackers of the system. We then applied this model to three attacker profiles we drafted based on interviews with DMS developers: the Cyber Criminal, the Service Competitor and the Animal Rights Activist. In a second set of interviews with DMS developers we determined and ranked the specific threats that each attacker could potentially pose. This resulted in identifying that attacks on the system's data confidentiality and data integrity are most likely to occur. The next step in our research was to find an encryption algorithm that suits this DMS given the constraints and identified threats. We evaluated the performance of different encryption algorithms with varying parameters. Our main finding was that the usage of hardware accelerated encryption operations results in higher performance with less energy consumption. In terms of security, `AES-256` with the `CCM` mode of operation protects against attacks on data confidentiality and data integrity, while still drawing less than 5 mAh over ten years of usage. To round off the system design, we implemented and evaluated a key exchange scheme so that Animal Trackers are able to dynamically negotiate a secure encryption key with a PU. The scheme we have evaluated is a pairing-free identity-based authenticated key agreement protocol (PF-IDAKA) that requires the exchange of only two messages. The main requirements for the key agreement are that it draws as little energy as possible and that it runs quickly. The speed of the protocol is important, because it directly influences the convenience of using the scheme in practice. This evaluation also considered the performance difference between a software and a hardware-accelerated approach. We found that there is little practical difference between the two in terms of power consumption. Considering that the protocol is generally run only once during the lifetime of an Animal Tracker, both options are viable. However, when looking at the speed of both protocol versions, the hardware-accelerated variant performed significantly better.

To summarize, we recommend usage of the PF-IDAKA scheme to set up symmetric keys between Animal Trackers and a PU and then switch to using `AES-256` with the `CCM` mode of operation for the actual encryption of data. For the PF-IDAKA, a 512-bit elliptic curve should be used, as this matches the security level that `AES-256` provides. If hardware acceleration is used for both the key agreement and the long-term data encryption, this system can offer confidentiality and integrity of data without consuming significant additional energy. Moreover, it does not require a long or complicated set up, but can instead be switched on within seconds.

References

[1] K. L. Lueth, "IoT 2019 in review: The 10 most relevant IoT developments of the year," IoT Analytics, January 2020, [Date retrieved: 10-12-2020]. [Online]. Available: https://iot-analytics.com/iot-2019-in-review/

[2] Statista, "Data volume of internet of things (IoT) connections worldwide in 2018 and 2025," June 2019, [Date retrieved: 10-12-2020]. [Online]. Available: https://www.statista.com/statistics/1017863/worldwide-iot-connected-devices-data-size/

[3] P. McDaniel and S. McLaughlin, "Security and privacy challenges in the smart grid," *IEEE Security Privacy*, vol. 7, no. 3, pp. 75–77, 2009.

[4] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[5] H. Groenendaal, D. T. Galligan, and H. A. Mulder, "An economic spreadsheet model to determine optimal breeding and replacement decisions for dairy cattle," *Journal of dairy science*, vol. 87, no. 7, pp. 2146–2157, 2004.

[6] J. Fruhlinger, "Top cybersecurity facts, figures and statistics for 2020," March 2020, [Date retrieved: 10-12-2020]. [Online]. Available: https://www.csoonline.com/article/3153707/top-cybersecurity-facts-figures-and-statistics.html

[7] R. Triggs, "Fact check: Is smartphone battery capacity growing or staying the same?" Android Authority, July 2018, [Date retrieved: 10-12-2020]. [Online]. Available: https://www.androidauthority.com/smartphone-battery-capacity-887305/

[8] E. Nilsson and C. Svensson, "Power Consumption of Integrated Low-Power Receivers," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 4, pp. 273–283, Sep. 2014.

[9] W. Xiong and R. Lagerström, "Threat modeling – a systematic literature review," *Computers and Security*, vol. 84, pp. 53–69, 2019.

[10] T. Xin and B. Xiaofang, "Online banking security analysis based on STRIDE threat model," *International Journal of Security and its Applications*, vol. 8, no. 2, pp. 271–282, 2014.

[11] J. Sanfilippo, T. Abegaz, B. Payne, and A. Salimi, "STRIDE-Based threat modeling for MySQL databases," *Advances in Intelligent Systems and Computing*, vol. 1070, pp. 368–378, 2020.

[12] M. Georgescu, H. Hazeyama, T. Okuda, Y. Kadobayashi, and S. Yamaguchi, "The STRIDE towards IPv6: A comprehensive threat model for IPv6 transition technologies," 2016, pp. 243–254.

[13] A. Omotosho, B. Ayemlo Haruna, and O. Mikail Olaniyi, "Threat modeling of Internet of Things health devices," *Journal of Applied Security Research*, 2019.

[14] A. Lenin, J. Willemson, and D. P. Sari, "Attacker Profiling in Quantitative Security Assessment Based on Attack Trees," in *Secure IT Systems*, ser. Lecture Notes in Computer Science, K. Bernsmed and S. Fischer-Hübner, Eds. Springer International Publishing, 2014, pp. 199–212.

[15] A. Jürgenson and J. Willemson, "On Fast and Approximate Attack Tree Computations," in *Information Security, Practice and Experience*, J. Kwak, R. H. Deng, Y. Won, and G. Wang, Eds. Springer Berlin Heidelberg, 2010, pp. 56–66.

[16] R. Dantu, P. Kolan, and J. Cangussu, "Network risk management using attacker profiling," *Security and Communication Networks*, vol. 2, no. 1, pp. 83–96, 2009.

[17] A. A. Cardenas, T. Roosta, and S. Sastry, "Rethinking security properties, threat models, and the design space in sensor networks: A case study in SCADA systems," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1434–1447, Nov. 2009.

[18] A. Filippoupolitis, G. Loukas, and S. Kapetanakis, "Towards real-time profiling of human attackers and bot detection," p. 6, 2014.

[19] B. Schneier, *Secrets and Lies.* John Wiley and Sons, 2000.

[20] "Specification for the advanced encryption standard (aes)," Federal Information Process in Standards Publication 197, 2001, [Date retrieved: 10-12-2020]. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[21] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, Feb. 1978.

[22] R. Kramer and V. Shoup, "Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack," *SIAM Journal on Computing*, vol. 33, no. 1, pp. 167–226, 2019.

[23] D. Kwon, J. Kim, S. Park, S. H. Sung, Y. Sohn, J. H. Song, Y. Yeom, E.-J. Yoon, S. Lee, J. Lee, S. Chee, D. Han, and J. Hong, "New Block Cipher: ARIA," in *Information Security and Cryptology - ICISC 2003*, J.-I. Lim and D.-H. Lee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 432–445.

[24] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Fast Software Encryption*, R. Anderson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 191–204.

[25] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms — design and anal-

ysis," in *International Workshop on Selected Areas in Cryptography*. Springer, 2000, pp. 39–56.

[26] D. J. Bernstein, "ChaCha, a variant of Salsa20," in *Workshop Record of SASC*, vol. 8, 2008, pp. 3–5.

[27] R. M. Needham and D. J. Wheeler, "Tea extensions," *Report, Cambridge University*, 1997.

[28] "Block cipher mode of operation," Wikipedia, [Date retrieved: 10-12-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[29] D. W. Davies and G. I. P. Parkin, "The average cycle size of the key stream in output feedback encipherment," in *Cryptography*, T. Beth, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 263–279.

[30] J. Jonsson, "On the security of ctr+ cbc-mac," in *International Workshop on Selected Areas in Cryptography*. Springer, 2002, pp. 76–93.

[31] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (gcm) of operation," in *International Conference on Cryptology in India*. Springer, 2004, pp. 343–355.

[32] P. Rogaway, "Evaluation of some blockcipher modes of operation," 2011.

[33] S. Bellovin, "Problem Areas for the IP Security Protocols," in *USENIX Security Symposium*, 1996.

[34] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[35] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*. Springer, 1985, pp. 417–426.

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

[37] C. Kerry and P. Gallagher, "FIPS PUB 186-4: Digital Signature Standard (DSS)," National Institute of Standards and Technology, 2013.

[38] D. R. Brown, "SEC 1: Elliptic curve cryptography," Standards for Efficient Cryptography, 2009.

[39] ——, "SEC 2: Recommended elliptic curve domain parameters," Standards for Efficient Cryptography, 2010.

[40] M. Campagna, "SEC 4: Elliptic curve Qu-Vanstone implicit certificate scheme (ECQV)," Standards for Efficient Cryptography, 2013.

[41] S. Blake-Wilson, N. Bolyard, V. Gupta, C. Hawk, and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," May 2006, [Date retrieved: 10-12-2020]. [Online]. Available: https://www.rfc-editor.org/info/rfc4492

[42] P. Wuille, "BIP32: Hierarchical deterministic wallets," GitHub, 2012, [Date retrieved: 10-12-2020]. [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki

[43] T. Güneysu, C. Paar, and J. Pelzl, "On the security of elliptic curve cryptosystems against attacks with special-purpose hardware," *Special-Purpose Hardware for Attacking Cryptographic Systems–SHARCS'06*, pp. 03–04, 2006.

[44] "Elliptic curve," Wikipedia, [Date retrieved: 10-12-2020]. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic_curve

[45] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[46] M. Ramdani, M. Benmohammed, and N. Benblidia, "Comparison of scalar point multiplication algorithms in a low resource device," *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 4, pp. 425–432, 2020.

[47] D. Mahto, D. A. Khan, and D. K. Yadav, "Security analysis of elliptic curve cryptography and RSA," in *Proceedings of the world congress on engineering*, vol. 1, 2016, pp. 419–422.

[48] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Workshop on the theory and application of cryptographic techniques*. Springer, 1984, pp. 47–53.

[49] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Annual international cryptology conference*. Springer, 2001, pp. 213–229.

[50] X. Cao, W. Kou, and X. Du, "A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges," *Information Sciences*, vol. 180, no. 15, pp. 2895–2903, 2010.

[51] S. H. Islam and G. Biswas, "An improved pairing-free identity-based authenticated key agreement protocol based on ECC," *Procedia Engineering*, vol. 30, pp. 499–507, 2012.

[52] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2681–2691, 2015.

[53] C. Kudla and K. G. Paterson, "Modular security proofs for key agreement protocols," in *International conference on the theory and application of cryptology and information security*. Springer, 2005, pp. 549–565.

[54] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM conference on Computer and communications security*, 1993, pp. 62–73.

[55] E. Barker, W. Barker, W. Burr, W. Polk, M. Smid *et al.*, *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration, 2020.

A Appendices

A.1 Threat Metrics Assessment

Paper	Metric	Relevance	Quantifiable	Comments
Lenin et al.	Budget	High	Yes	Monetary resources often define limits on the scale of possible attacks.
	Skills	High	Yes	An attacker's technical skills directly influence the potential complexity of his attacks.
	Time	High	Yes	Time constraints place an upper limit on attack complexity, as complex attacks often require more time.
Dantu et al.	Budget	-	-	-
	Skills	-	-	-
	Time	-	-	-
	Risk	Medium	Yes	An attacker who is willing to take risks might perform attacks that safer attackers will not attempt.
	Perseverance	Medium	Yes	A persistent attacker might attempt different types of attacks and therefore poses a greater threat.
	Tenacity	-	-	See 'Perseverance'.
	Motives	High	No	An attacker's motive depends on a lot of factors and cannot be quantified.

Table A1: The metrics that were extracted from the first two papers. Every metric has been assessed based on its relevance and its quantifiability. Some metrics appear multiple times. To save space and time, duplicate metrics were only assessed once. Such duplicates entries are denoted with an empty row.

Paper	Metric	Relevance	Quantifiable	Comments
Cardenas et al.	Budget	-	-	-
	Physical access	High	Yes	Whether or not an attacker can gain physical access to the resource he is attacking has a large impact on the types of attacks he can perform.
	Skills	-	-	-
Filippoupolitis et al.	Skills	-	-	-
	Education	Medium	Yes	Level of education can describe other attacker properties, such as his approach to performing attacks.
	Risk	-	-	
	Gender	Low	Yes	An attackers gender does not influence how much of a threat they pose.
	Goal	High	No	The type of an attack is often dictated by its goal.
	Speed	Low	Yes	Irrelevant in this context (especially given that 'Time' is already considered).
	Mistakes	Low	Yes	Irrelevant in this context.
	Anti-forensics	Medium	No	Going unnoticed during an attack crosses out certain attacks, e.g. ones that are highly observable.
	Success	Low	Yes	Irrelevant in this context.

Table A2: The metrics that were extracted from the last two papers. Every metric has been assessed based on its relevance and its quantifiability. Some metrics appear multiple times. To save space and time, duplicate metrics were only assessed once. Such duplicates entries are denoted with an empty row.

A.2 Attacker Profile Scoring

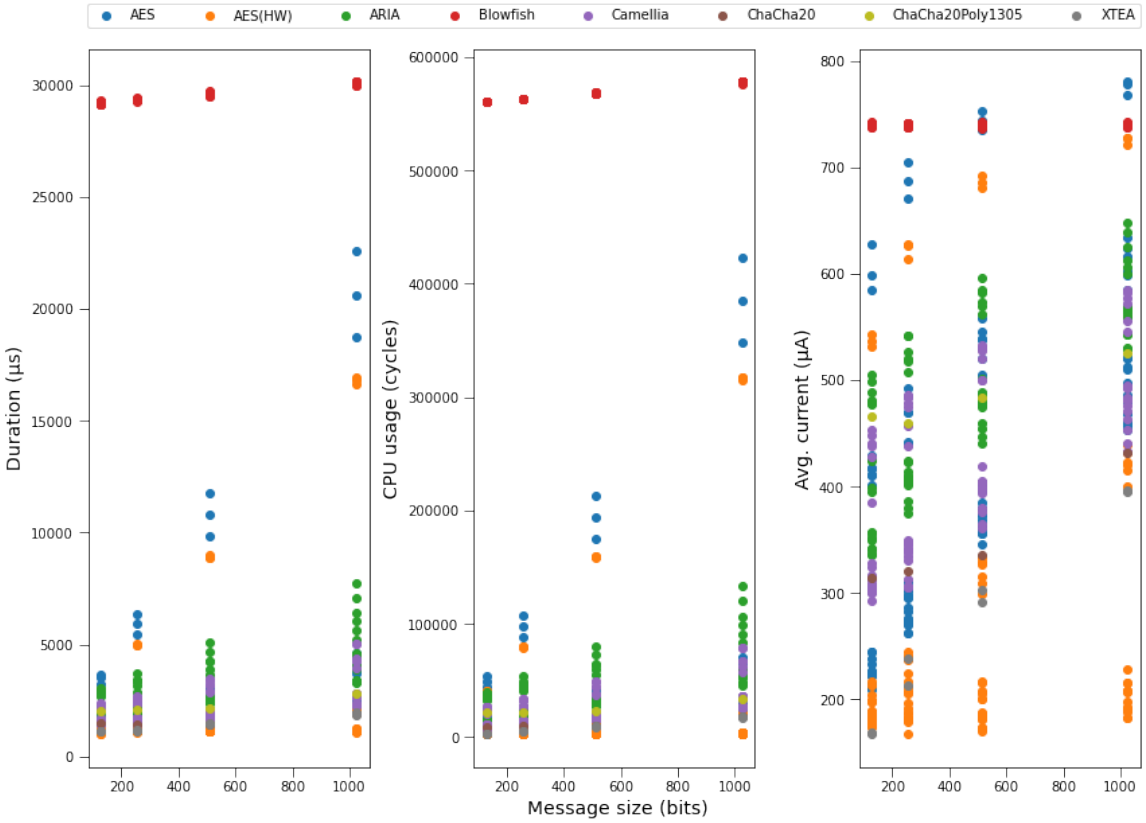
We have conducted individual interviews with three DMS developers working in different technical fields. We first constructed the three attacker profile outlines (Cyber Criminal, Service Competitor and Animal Rights Activist) and then discussed these outlines with the three employees in order to fill them with actual characteristics. We then asked the employees to estimate a score for each of seven quantitative metrics, for every attacker profile. They were also asked to provide potential motives and goals for the attacker profiles, based on their own experiences and intuition. The table below shows the scores that each of the employees estimated for the different metrics.

Profile	**Metric**	**Empl. 1**	**Empl. 2**	**Empl. 3**	***Average***
Cyber Criminal	Budget	5	4	6	5
	Skills	10	10	10	10
	Time	7	6	8	7
	Physical access	0	0	5	1.67
	Risk	8	7	7	7.33
	Perseverance	8	7	9	8
	Education	7	9	8	8
Service Competitor	Budget	7	9	7	7.67
	Skills	6	8	8	7.33
	Time	9	9	9	9
	Physical access	5	0	9	4.67
	Risk	3	5	4	4
	Perseverance	8	7	8	7.67
	Education	8	9	9	8.67
Animal Rights Activist	Budget	6	7	4	5.67
	Skills	7	6	5	6
	Time	7	8	8	7.67
	Physical access	10	5	6	7
	Risk	9	9	5	7.67
	Perseverance	9	7	7	7.67
	Education	7	7	6	6.67

A.3 Symmetric Cipher Benchmark Results

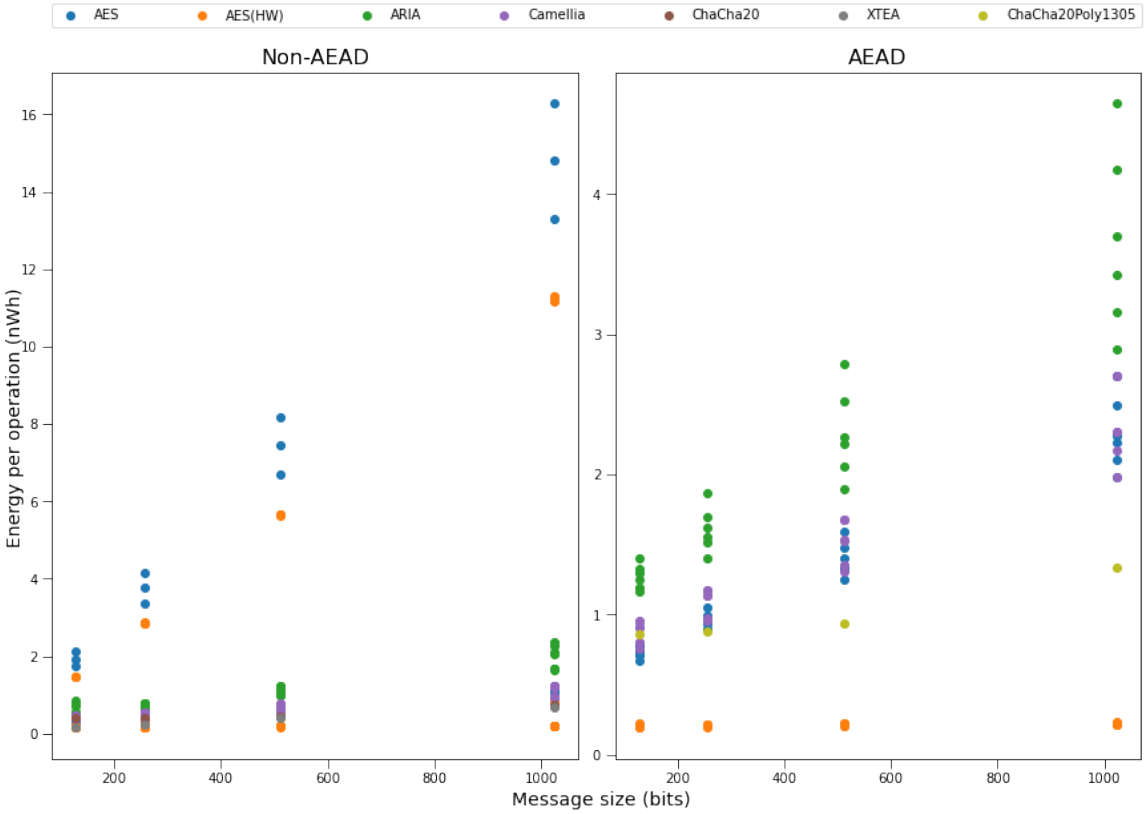
A.3.1 Complete Overview

A scatter plot showing an overview of all measured performance metrics. Each point represents a unique cipher-mode-key combination. This plot clearly shows that `Blowfish` (red) performs the worst in two out of three performance metrics by a large margin, and is close to being the worst in the third.



A.3.2 AEAD versus non-AEAD

A scatter plot showing the difference in energy consumption of AEAD and non-AEAD encryption. For most ciphers (distinguishable by the different colors), the average energy consumption increases when authenticated encryption is used. Only `AES(HW)` remains practically constant.



A.3.3 Influence of Key and Message Sizes on Performance Metrics

The following graphs show how, on average, the three performance metrics are influenced by increasing key and message sizes. The graphs were obtained by grouping all data samples by their key or message size, respectively.