Cryptography for a Networked Control System using Asynchronous Event-Triggered Control

P.Zijlstra

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Cryptography for a Networked Control System using Asynchronous Event-Triggered Control

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

P.Zijlstra

March 1, 2016

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Networked control systems (NCSs) are control systems in which the control loops are closed through a communication network. These systems usually consist of sensors, actuators, controllers and communication devices. The disruption of these control systems can have a significant impact on public health, safety and lead to large economic losses, e.g. the electric power distribution, natural gas distribution and transportation systems. This indicates the importance of protecting the NCS against a malicious party which is attacking the cyber infrastructure. Hence, the NCS should be able to prevent or survive the attacks attempted by the malicious party.

This thesis examines an NCS using a recently presented asynchronous event-triggered implementation. This implementation only uses one bit for each transmission, which largely reduces the total number of transmissions. Implementing an aperiodic controller means that sensor and control communications are limited to instances when the system needs attention. Due to the fact that transmissions only take place when control action is needed, it is imperative that these are not tampered with by a malicious party.
Considering the principle of an asynchronous event-triggered based NCS, a security strategy is presented which can deal with eavesdropping, data modification and compromised-key attacks. The strategy consists of three main parts: encryption of the single bit sampling transmissions, key updates to spare the transmitted packages' length and the injection of dummies into the wireless network.

Contents

List of Figures

List of Tables

Preface

This document is a part of my Master of Science graduation thesis. The idea of doing my thesis on this subject came after an elective course I followed during the first year of the MSc program. The course was called; Security and Cryptography and was given by Dr.ir. J.C.A. van der Lubbe. I became very interested in cryptography during the course of the lectures. My idea was to investigate a subject in which control could be combined with cryptography, but at the time, I did not know if this was possible or if these subjects were too far apart.
The MSc coordinator, who to my surprise and happiness was very open to the idea, suggested meeting with different professors. The meetings with these professors led me to my supervisor Dr.ir. M.Mazo.Jr. He was very open to combine control with cryptography and already had some ideas for a topic. This allowed me to pursue the subject at hand, which combines cryptography and networked control.

Acknowledgements

I would like to thank my supervisors Dr.ir. M.Mazo.Jr. and A.Fu, MSc for their assistance during the writing of this thesis. Their guidance is much appreciated, for otherwise I would be lost in a forest of papers.
Furthermore, I really liked the help from the MSc.Coordinator and the professors at DCSC in supporting my idea for a thesis topic. Because of this help, I was able to investigate a subject which combines control and cryptography.

Delft, University of Technology P.Zijlstra
March 1, 2016

Sherlock Holmes about encrypted messages:

> “It is obviously an attempt to convey secret information."
> – *"The Valley of Fear"*, A.C. Doyle

Chapter 1

Introduction

A Networked Control System (NCS) is a control system in which the control loop is closed through a communication network. An NCS usually consists of sensors, actuators, controllers and communication devices. NCSs merge information technology, sensor technology and communication technology, which makes them widely used in modern control systems. As a consequence, the security of these types of control systems is very important. A recently presented asynchronous event-triggered implementation in [3] only uses one bit for each transmission, which largely reduces the total number of transmissions. However, this implementation is not robust in the face of attacks or faults, which largely increases the difficulty of the design of the security strategy. This thesis considers the security issue of an NCS with the aforementioned asynchronous event-triggered mechanism. A proposal for the security strategy follows from the consideration of the different aspects related to the security issue.

1-1 Motivation

NCSs are applied in a large variety of situations; e.g. electric power distribution, natural gas distribution and transportation [4, 5]. Hence, NCSs can have a significant impact on public health, safety and security. The disruption of these systems can lead to large economic losses or even personal injuries. There are multiple examples of attacks on NCSs with disastrous consequences, such as the Maroochy water breach [6, 7] and the wireless carjacking of a Jeep [8, 9].
In the year 2000, the wireless control system of Maroochy Water Services company in Australia was hacked. The wireless communications with sewage pumping stations got lost, pumps were not working correctly and the alarms put in place to notice malfunctions did not go off. This caused 800,000 liters of sewage to discharge into local parks, rivers

and onto the grounds of a hotel. As a consequence marine life in the area died, creek water turned black and the sewage caused a horrible stench for the local inhabitants.
In July 2015, hackers broke into the wireless control system of a Jeep and remotely took over its controls. This gave them access to control several parts of the vehicle, e.g the radio, windscreen wipers and apply the wiper fluid. Even more concerning is the fact that at lower speeds, it was possible to disable the brakes and turn the engine off. These examples indicate the necessity of protecting NCSs against attacks on the cyber infrastructure.

Nevertheless, at the moment most studies regarding the security of NCSs concentrate on the robustness of the controllers from the perspective of control theory. In [10], a network packet scheduling attack on the system is considered. By changing the temporal characteristics of the network, this attack can lead to time-varying delays and changes in the order in which packets are delivered. A robust output-feedback controller is designed which uses the received packets in a minimax sense. Another example is [11], where NCSs under Denial-of-Service (DoS) attacks are considered. In a DoS attack, a large number of messages are sent with the aim of flooding the communication channel. A robust feedback controller is designed which minimizes an objective function, subject to safety and power constraints.
In the foregoing examples, the focus is on the NCS having a robust design, which renders the attacks ineffective. These kinds of strategies can be seen as reactive, anti-attack methods. This means that actions are taken when the attacks already had an effect on the NCS [4, 12, 13]. These actions include attack detection, isolation or correction, all of which are based on the robustness of the system. Thus the following problems could arise:

1. The reactions may not be efficient: not all kinds of attacks can be predicted.
2. The attacks still cause problems: the attacks have a minimal effect on the systems, such that these attacks cannot be detected and isolated.
3. The systems may not be optimized: there are some trade-offs between robustness, system stability and system performance.

Besides reactive, there are also proactive techniques such as encryption and dummy injection. These proactive techniques stem from cryptography. They do not have much influence on the system performance, but can be more efficient compared to the reactive strategy and create a higher level of security. Hence, some proactive actions are necessary to be introduced to NCSs to prevent attacks. The focus of this thesis is on how to encrypt the feedback formulation.

Following a newly presented Asynchronous Event-Triggered Control (AETC) strategy in [3], an NCS implementation is considered. By designing a threshold from the system's current state and decentralize it to each sensor, a decentralized event-triggered mechanism can be realized. With both controller and sensor sharing the same threshold, only one bit is needed for the transmissions of state sampling. Thus, the NCS needs less attention of the feedback channel compared to the centralized event-triggered implementation. The results in [14] support this conclusion. The reduced amount of attention makes the asynchronous

event-triggered strategy especially suitable for a wireless NCS, which always has a limited available bandwidth.

However, by using the AETC implementation, transmissions will only be sent along the feedback channel when necessary. As a result, the NCS is less robust than periodically controlled structures. If a single bit is changed or destroyed by an attack, the system can become functionless or disabled. Therefore, the implementation with an asynchronous event-triggered mechanism can easily be hacked. A strategy needs to be designed for this kind of implementation; to keep it safe and secured.

Standard encryption algorithms need a lot of processing power and can introduce delays, which is always unacceptable for real-time control systems. Therefore, it is necessary to develop a modified encryption algorithm, specifically suited for this NCS. It should satisfy the following constraints:

1. Safety: since the transmissions only use one bit, the algorithm should be safe and update its key frequently.

2. Limited message length: since the wireless NCS has a limited bandwidth, the messages should be as short as possible.

3. Fast: due to the real-time implementation, the encryption and decryption have a limited processing time.

1-2 Problem Statement

Considering the principle of an asynchronous event-triggered based NCS, a security strategy is presented which can deal with the following attack types: eavesdropping, data modification and compromised-key attacks. This strategy consists of three sub-strategies, namely:

1. Sampling encryption: use keys to encrypt the single bit sampling transmissions.

2. Key update: update the keys frequently to spare the transmitted packages' length, while assuring a certain level of security.

3. Dummy injection: inject dummies into the wireless network to hide the useful messages.

The main problem to solve in this thesis is to design algorithms regarding these sub-strategies.

Survey Structure

The rest of this survey is organised as follows. Chapter 2 discusses the necessary fundamentals. An overview of the security solution is given in chapter 3. Chapter 4 shows results of the simulations and this survey ends with a discussion and suggestions for future work in chapter 5.

Chapter 2

Preliminaries

This chapter reviews the fundamental subjects related to the security of the Networked Control System (NCS) under consideration. First, the system structure and the relevant security concepts will be presented. Followed by an introduction to the related source coding theories as well as the concept of binary distances. The chapter finishes with a discussion on the different ways to measure the state of the security.

2-1 System Structure

The employed control scheme is an Asynchronous Event-Triggered Control (AETC). The applied structure will now be described, starting with a brief introduction of the notations that are going to be used subsequently.

2-1-1 Asynchronous Event-Triggered Control

The positive real numbers are denoted by $\mathbb{R}^+$ and by $\mathbb{R}_0^+ = \mathbb{R}^+ \cup 0$. Natural numbers, including zero are denoted by $\mathbb{N}$. $|\cdot|$ denotes the Euclidean norm in the appropriate vector space, when applied to a matrix $|\cdot|$ denotes the l_2 induced matrix norm. For a set, $|\cdot|$ denotes the cardinality of this set. $\|\cdot\|_\infty$ is used for the $\mathcal{L}_\infty$ functional norm. A function α: $\mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{K}(\alpha \in \mathcal{K})$ if: α is a continuous function, $\alpha(0) = 0$ and $s_1 > s_2 \Rightarrow \alpha(s_1) > \alpha(s_2)$. A function α: $\mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{K}_\infty(\alpha \in \mathcal{K}_\infty)$ if: $\alpha \in \mathcal{K}$ and $\lim_{s\to\infty} \alpha(s) = \infty$. A function α: $\mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{L}(\alpha \in \mathcal{L})$ if: α is a continuous function, $s_1 \geq s_2 \Rightarrow \alpha(s_1) \leq \alpha(s_2)$ and $\lim_{s\to\infty} \alpha(s) = 0$. A function α: $\mathbb{R}^+ \to \mathbb{R}^+$ belongs to class $\mathcal{KL}(\alpha \in \mathcal{KL})$ if: $\forall(\text{fixed})t : \beta(\cdot, t) \in \mathcal{K}$ and $\forall(\text{fixed})s : \beta(s, \cdot) \in \mathcal{L}$.

Definition 2-1.1. *(Asymptotic Stability)[15]*
A system $\dot{\xi}(t) = f(\xi(t))$, $t \in \mathbb{R}_0^+$, $\xi(t) \in \mathbb{R}^n$ is said to be uniformly globally asymptotically stable (UGAS) if there exists $\beta \in \mathcal{KL}$ such that for any $t_0 \geq 0$ the following holds:

$$\forall \xi(t_0) \in \mathbb{R}^n, |\xi(t)| \leq \beta(|\xi(t_0)|, t - t_0), \forall t \geq t_0.$$

Definition 2-1.2. *(Input-to-State Stability)[15]*
A control system $\dot{\xi} = f(\xi, v)$ is said to be (uniformly globally) input-to-state stable (ISS) with respect to v if there exists $\beta \in \mathcal{KL}$, $\gamma \in \mathcal{K}_\infty$ such that for any $t_0 \in \mathbb{R}_0^+$ the following holds:

$$\begin{aligned} \forall \xi(t_0) \in \mathbb{R}^n, \|v\|_\infty < \infty \Rightarrow \\ |\xi(t)| \leq \beta(|\xi(t_0)|, t - t_0) + \gamma(\|v\|_\infty), \forall t \geq t_0. \end{aligned}$$

The ISS property of a system can also be established by means of ISS-Lyapunov functions, i.e. a system is ISS if and only if a smooth ISS-Lyapunov function exists [15].

Definition 2-1.3. *(ISS Lyapunov function)[15]*
A continuously differentiable function $V: \mathbb{R}^n \to \mathbb{R}_0^+$ is said to be an ISS Lyapunov function for the closed-loop system $\dot{\xi} = f(\xi, v)$ if there exists class $\mathcal{K}_\infty$ functions $\underline{\alpha}$, $\overline{\alpha}$, α_v and α_e such that for all $\xi \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$ the following is satisfied:

$$\begin{aligned} \underline{\alpha}(|\xi|) \leq V(\xi) \leq \overline{\alpha}(|\xi|), \\ \nabla V \cdot f(\xi, v) \leq -\alpha_v \circ V(\xi) + \alpha_e(|v|). \end{aligned} \tag{2-1.1}$$

Now follows the non-linear system that is going to be analysed. Consider a non-linear system of the form

$$\dot{\xi}(t) = f(\xi(t), v(t)), \forall t \in \mathbb{R}_0^+, \tag{2-1.2}$$

where $\xi(t) \in \mathbb{R}^n$, $v(t) \in \mathbb{R}^m$, f is locally Lipschitz. Assume a locally Lipschitz controller $k : \mathbb{R}^n \to \mathbb{R}^m$ is available, rendering the closed-loop system ISS with respect to measurement errors ε:

$$\dot{\xi}(t) = f(\xi(t), k(\xi(t) + \varepsilon(t))). \tag{2-1.3}$$

Furthermore, assume a sample-and-hold implementation for the controller:

$$v(t) = k(\hat{\xi}(t)), \tag{2-1.4}$$

where

$$\begin{aligned} \hat{\xi}(t) = [\hat{\xi}_1(t), \hat{\xi}_2(t), \cdots, \hat{\xi}_n(t)]^{\mathrm{T}}, \\ \hat{\xi}_i(t) = \xi_i(t_{r_i}^i), t \in [t_{r_i}^i, t_{r_i+1}^i), \forall i = 1, \cdots, n. \end{aligned} \tag{2-1.5}$$

Representing the sample-and-hold effect as a measurement error, gives:

$$\varepsilon_i(t) := \hat{\xi}_i(t) - \xi_i(t).$$

The sequences of local sampling times $\{t^i_{r_i}\}$ for each sensor i are determined by the triggering condition:

$$t^i_{r_i} = \min\{t > t^i_{r_i-1} | \varepsilon_i^2(t) = \eta_i(t)\}, \tag{2-1.6}$$

where $\eta_i(t)$ is a local threshold. These local thresholds are determined by the global threshold $\eta(t)$ and a predesigned distributed parameter θ_i:

$$\eta_i(t) = \theta_i^2 \eta(t)^2, |\theta| = 1. \tag{2-1.7}$$

The initial value of the global threshold $\eta(t_0)$ is restricted by the initial state of the system as:

$$\eta(t_0) \geq \frac{\mu}{\rho} \underline{\alpha}^{-1} \circ V(\xi(t_0)), \tag{2-1.8}$$

where ρ is a design parameter that must satisfy $\underline{\alpha}^{-1} \circ \overline{\alpha}(\underline{\alpha}^{-1} \circ \epsilon \alpha_v^{-1} \circ \alpha_e(s) + 2s) \leq \rho s$ for all $s \in (0, \eta(t_0)]$ and some $\epsilon > 0$; and $\mu \in (0.5, 1)$ is a predesigned parameter that determines how the threshold is updated:

$$\eta(t^c_{r_c+1}) = \mu \eta(t^c_{r_c}). \tag{2-1.9}$$

The threshold is updated at times $t^c_{r_c+1}$:

$$t^c_{r_c+1} = \min\{t = t^c_{r_c} + r\tau^c | r \in \mathbb{N}^+, |\overline{\xi}(t)| \leq \overline{\alpha}^{-1} \circ \underline{\alpha}(\rho \eta(t^c_{r_c}))\}, \tag{2-1.10}$$

where $|\overline{\xi}(t)| := |\hat{\xi}(t)| + \eta(t)$ and τ^c is a design parameter at which the sensors need to switch into listening mode to receive potential threshold update signals. Given that the controller knows which threshold $\eta_i(t)$ is being employed by each sensor, only one bit is needed from the sensors to indicates that an event occurred. This bit contains the sign of the error, i.e.

$$\hat{\xi}_i(t^i_{r_i}) = \hat{\xi}_i(t^i_{r_i-1}) + \text{sign}(\varepsilon_i(t^i_{r_i}))\sqrt{\eta_i(t^i_{r_i})}. \tag{2-1.11}$$

Finally, in [3] it is shown that this implementation renders the resulting closed-loop UGAS.

2-1-2 Network Protocol

Assume this control system has its actuator and all its sensors distributed on a large physical scale. Figure 2-1 depicts a description of the sensor network under consideration. Each sensor is co-located with a wireless node and a processor, thus forming a sensor node. There is one controller in this system, which also acts as the central node of the network. The controller and processors are used to encrypt/decrypt transmissions, inject/filter dummies and perform key updates. The controller is co-located with the actuator, which calculates the inputs and forwards them to the actuator after decrypting the information from the sensor nodes.

The focus of this thesis will be on the security of a scalar problem. More specifically, on the communications from the sensor towards the controller. These communications require only the exchange of one bit of information for every sensor update and thus only require

a small payload. It is preferred that the encrypted message length is as small as possible to retain the small payload. This in order to save the battery life of the sensors and the actuator. However the single bit communication is not robust in the face of an attack or faults, since only 1 altered bit can result in a functionless system. Therefore encryption and source coding techniques will be applied to improve the security and the tolerance for faults. Since a transmission only occurs in case of an event, this fact would also be clear for an eavesdropping attacker. To obscure the occurrence of an event, dummy messages will be added to the communication channel.
Most of these techniques will increase the length of the message. This means that a balance has to be found between the message length and the level of security.

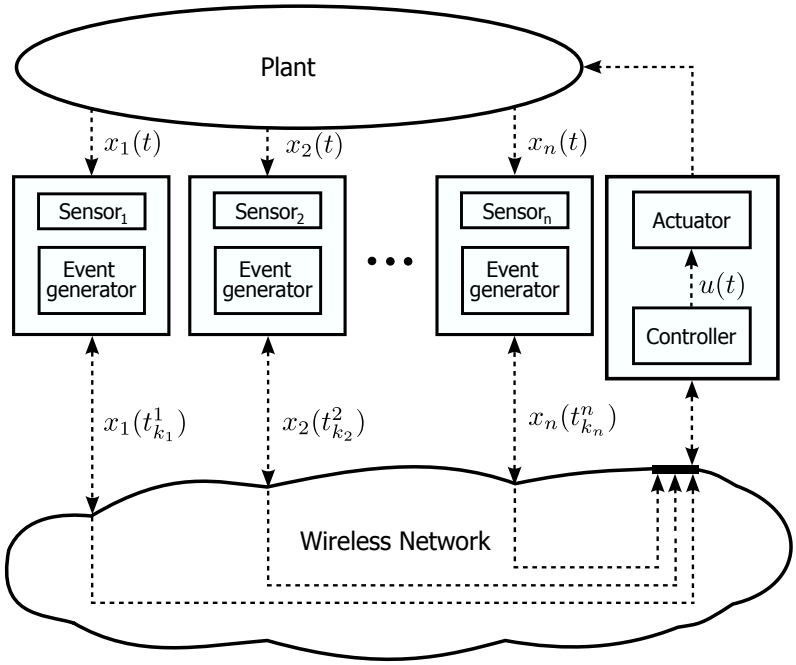

Figure 2-1: Networked system structure

2-2 Security Concepts

In order to improve the current security state of the NCS; several concepts from cryptography and source coding are used. The relevant concepts will be discussed next and are followed by an exposition of security measures.

2-2-1 Cryptography

Cryptography is the practice and study of techniques for secure communication in the presence of third parties (attackers) [16]. It is concerned with techniques based on a secret key used for concealing or enciphering data. Only someone with access to the key is capable of deciphering the encrypted information. In essence, it is impossible for anyone else to do so.
Depending on the situation, there can be different purposes for employing encryption. In this case, the one-bit communication between sensor and controller has to be securely transmitted and even in the presence of an attacker, the system must remain secure.
A simple encryption scheme is presented in figure 2-2 to explain the basic idea behind encryption. Assume the sensor wants to communicate a message M with the controller. In this case the message can be '0','1' or a dummy. A key K is derived beforehand, by using e.g. a Key Derivation Function (KDF) to obtain a pseudorandom key and is distributed to the sensor and the controller. To be able to send an encrypted message, the sensor encrypts M by using K as input for the encryption algorithm to create the encrypted message (or ciphertext) C. When C is received by the controller, it is able to obtain the original message by using the decryption algorithm and the same key.

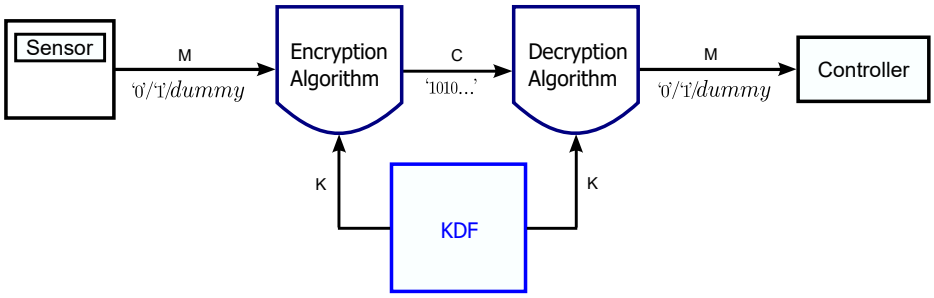

Figure 2-2: Encryption scheme example

Key Derivation Function

The key(s) being used for the communications must be as unpredictable as possible. If certain keys were to occur with a higher probability than others, a potential intruder would find it easier to discover the used key. This unpredictability of the keys can be ensured by using a pseudo-random generator such as the DES algorithm or a shift register [17], which generates a pseudo-random 64 bit pattern. In the security strategy for the NCS, the current samplings can be used as the source of the keys.

Cryptoperiod

The cryptoperiod is described in [18] as the time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect.

This cryptoperiod can be defined by an arbitrary time period or the maximum amount of data protected by the key [18]. During the determination of a cryptoperiod, the risk and consequences of exposure have to be considered. Keys used to protect the confidentiality of communications usually have shorter cryptoperiods than keys used to protect stored data. Cryptoperiods are generally longer for stored data, because the overhead of a key update may be cumbersome [18].

The cryptoperiod can be designed based on the following properties [17]:

- *Sensitivity of enciphered data*: the more important it is to guarantee a high level of security, the more frequently a key will be exchanged for a new one.
- *Period of validity of the protected data*: the validity of messages which are communicated. At the time of transmission, the messages may contain valuable information for an intruder. However, a few moments later the messages can becomen insignificant.
- *Strength of the cryptographic algorithm*: The first measure of strength of an encryption algorithm stems from the resulting key length. The length of the key determines the time it will take for a brute force attack to find the correct key. The second measure is the strength of the encryption algorithm itself. Due to cryptanalytic methods, the key can be found in a smaller timespan than by using an exhaustive key search. This time span is different for each encryption algorithm.

A shorter cryptoperiod can enhance the security of the NCS, because a specific key will be exposed to sniffers for a shorter period of time. However, a shorter cryptoperiod means more frequent key changes, which increases the risk of exposing the key generating algorithm. At the same time, a shorter cryptoperiod also means more consumption of system resources and a higher probability of faults while changing the keys. This trade-off should be considered while designing the cryptoperiod.

In [2] a model is proposed to obtain the cryptoperiod (T_c) and the minimim key length (n_m). The attacker is assumed to attempt an exhaustive key search in order to decrypt the ciphertext. Trying all possibilities means that with a key length n, 2^n operations may have to be performed. However using more efficient cryptanalitic methods, this value can be reduced by a few bits [19]. The estimate given in [2], is a decrease of 1 bit every 1,5 years. Meaning that after e.g. 3 years 2^{n-2} operations have to be performed. Another aspect which is taken into account is the fact that computation power increases as time goes by. Moore's law states that computation power doubles each 1.5 years. Attackers will therefore have an increasing amount of computation power available. This model is derived in the following manner:
The total amount of computation performed E, over a period of time t_c and a fixed computation power c equals: $E = ct_c$. The total time of an exhaustive key search is proportional to 2^n: $2^n = ct_c$. Hence, $t_c = \frac{2^n}{c}$. The number of bits of the security level follows from this formula and is shown in equation 2-2.1.

$$n = log_2(ct_c). \tag{2-2.1}$$

Following [2], a correction factor $g := 2^{t_c/1.5}$ has to be applied to the computation power to incorporate Moore's law. The estimated computation power available in a specific year follows from:

$$c = g \cdot c_e, \tag{2-2.2}$$

with c_e the current computation power. Computational power gain accumulated from the present moment up to t_c years can be found by integrating g over the period $[0, t_c]$, multiplied by c_e. The total amount of computation performed in t_c years is given by:

$$T = c_e(1.5/ln(2))(2^{(t_c/1.5)}) - 1) \tag{2-2.3}$$

The number of operations of the exhaustive key search equals: 2^n, hence the cryptoperiod and key length follow from inserting $T = E = 2^n$ and solving for t_c and n respectively.

$$t_c = 1.5 \cdot log_2 \left(\left(\frac{(2^n ln(2))}{(1.5c)} \right) + 1 \right) \tag{2-2.4}$$

$$n = log_2 \left(\frac{1.5c}{ln(2)} \right) + log_2(2^{(t_c/1.5)} - 1) \tag{2-2.5}$$

When the cryptanalytic advances are also taken into account ($E = 2^{n-(t_c/1.5)}$), the key length becomes:

$$n = log_2 \left(\frac{1.5c}{ln(2)} \right) + log_2(2^{(2t_c/1.5)} - 2^{(t_c/1.5)}) \tag{2-2.6}$$

It is not possible to solve this equation for t_c. However, it is possible to approximate the term: $2^{(2t_c/1.5)} - 2^{(t_c/1.5)} = 2^{(t_c/1.5)}(2^{(t_c/1.5)} - 1) \approx 2^{(t_c/1.5)}2^{(t_c/1.5)} = 2^{(2t_c/1.5)}$. This makes it possible to approximate the total search time for variable key length and available computation power:

$$t_c = \frac{1.5n}{2} - \frac{1.5}{2} log_2 \left(\frac{1.5c}{2} \right) \tag{2-2.7}$$

Where t_c is the amount of time that a key of size n is protected. This approximation can be used to calculate the cryptoperiod:

$$T_c = \frac{1.5n}{2} - \frac{1.5}{2} log_2 \left(\frac{1.5c}{2} \right). \tag{2-2.8}$$

Key Update Function

If the value of the new key is dependent on the value of the old key, the process is known as key update (i.e., the current key is modified to create a new key). The creation of a new key has to be accomplished by applying a one-way function to the old key and possibly additional data. Since a non-reversible function is used in the update process, previous keys are protected in the event that a key is compromised. However, future keys are not

protected. Key updates are often used to limit the amount of data protected by a single key. However, following the discussions in [18, 20], key updates cannot be used to replace a compromised key. The key used for the NCS will be updated once the cryptoperiod has passed or when there are no messages available to send i.e. all messages have already been transmitted. The latter is due to the fact that otherwise certain (encrypted) messages would be used multiple times. This would give a potential attacker more information on which messages correspond to a '0','1' or dummy message.

Dummy messages

Dummy messages are false messages sent with the purpose of concealing periods of inactivity and the useful information. The properties which dummy messages should satisfy are discussed in [21] . First, they should have the same binary structure as the real messages. This obscures which messages are valuable and which are dummies. Second, when analysing the dummies, the linguistic statistics should not give the attacker any extra information. The second property can be counteracted by ensuring that the messages have approximately the same probability of occurrence as any other bit sequences of the same length. In practice the messages are compressed to remove the redundancy present in the messages. Where the redundancy is equal to the total number of bits of the message, minus the number of bits of actual information present in the message.

2-2-2 Security

When considering the security issue of a system, there are three aspects that have to be considered [17]. The first aspect is the purpose of the security. Which in its turn can be subdivided into three categories: confidentiality, reliability and continuity. These categories are shown in table 2-1. This table also shows the way these aspects are handled for the communication between sensor and controller.
Dummy messages are used to conceal the fact that a message has been transmitted. Error detection and correction are used to deal with modified or distorted data. Finally, timestamps are applied to the transmissions to be able to deal with a replay attack [10]. In the replay attack, messages that have already been transmitted are reused and transmitted again by an attacker.

A second aspect is the dimension of the security. The dimension determines whether the security measures are designed for the prevention or the correction of the damage caused by an attack. For prevention, the one-bit message is encrypted to minimise the chance of an attacker compromising the transmitted information. Error correction is applied to the data transmissions to reduce bit errors caused by noise or an attacker. The key will be rendered useless when fallen into the hands of an attacker and a key update will be performed. When the KUF is compromised, a Re-key operation has to be performed, which is entirely independent of the value of the old key. To limit the amount of data protected by a single key, the key is updated after all messages are sent or the cryptoperiod has passed. When

Table 2-1: Purpose of the security

Category	Description	Security approach
Confidentiality	Reading or tapping of data	Dummy messages
Reliability	Modification of data or the corrosion of data files	Source coding: Error detection
Continuity	Distortion of data transmission or sending false data	Source coding: Error correction and timestamped transmissions

a key is compromised this will only give limited access to the attacker. A summary of the three dimension types is shown in table 2-2 .

Table 2-2: Dimension of the security

Category	Description	Security approach
Prevention	Minimise the chance of anything happening to the transmissions	Encryption
Correction	If anything happens with the encrypted data it must be possible to reconstruct or respond appropriately	Source coding: error correction Key update (or Re-key)
Damage limitation	When an attack occurs, ensure that the resulting damage remains as limited as possible	Key update (or Re-key)

The final aspect is the means of the security. The means can be subdivided into three parts as well: the security can be physical, organisational or hardware/software based. Physical security relates to the protection against the physical entry of an intruder. This can be accomplished by using e.g. metal containers, temperature or vibration sensors to prevent physical intrusion of the system. Here, the focus will be on the cryptographic algorithms and methods which fall into the hardware and software category. The organisational security provides conditions to allow the physical, hardware and software based security measures to be completely effective.

2-3 Source Coding

Source coding is used to transmit messages from a source to a recipient with a minimum amount of errors [22]. The wireless communication channels used for the communication between the sensor and controller are vulnerable to bit errors [23]. This section discusses several methods from source coding which are applied to detect and correct these errors.

2-3-1 Hamming Distance

The Hamming distance is a distance measure which can also be used for binary sequences. The formal definition of the Hamming distance is the distance between two strings f_s and g_s, with length n_s. This can be defined as [24]:

$$D_H(f_s, g_s) = \sum_{i=1}^{n_s} d_H(f_i, g_i) \tag{2-3.1}$$

where

$$d_H(f_i, g_i) = \begin{cases} 0, & f_i = g_i; \\ 1, & f_i \neq g_i; \end{cases} \tag{2-3.2}$$

The minimum distance d between two strings, with the Hamming distance as a distance measure equals

$$d = \min_{f_s \neq g_s} D_h(f_s, g_s). \tag{2-3.3}$$

In other words, the Hamming distance $D_H(f_s, g_s)$ represents the number of places where two different (bit) strings differ.

2-3-2 Minimum n from Source Coding Bounds

In source coding theory, several bounds exist regarding the relation between bit sequence length (n_b) and the number of messages (k_b) which can be placed a certain hamming distance (d_H) away from each other. Error coding is used to encode messages in such a manner that errors can be detected and when appropriate, corrected after transmission or storage [25]. An error code spreads the 2^{k_b} messages evenly over the total number of 2^{n_b} locations.
Every bit-error increases the required Hamming distance between the original and the corrupted messages by one. By adding a parity bit to the end of the message and if t_b errors need to be detected, this Hamming distance has to be: $d_{Hmin} > t_b$. A parity bit is used to check whether the message contains an even or an odd amount of ones. When errors also have to be corrected; this distance increases to $d_{Hmin} > 2t_b$.

Hamming Bound

The Hamming bound is a limit on how many bits of redundancy (n_b - k_b) have to be be added to a message to create a t_b-error correcting code [25]. A message sequence of length n_b has $\frac{n_b!}{d_H!(n_b-d_H)!}$ messages that are d_H bits away from it. The Hamming bound for a t_b-error correcting code is: $2^{k_b} + 2^{k_b} \times \sum_{d=1}^{t_b} \frac{n_b!}{d!(n_b-d)!} \leq 2^{n_b}$. Correcting codes which satisfy the Hamming bound exactly are called perfect codes. These perfect codes decode all received messages, whether they are valid or contain an error. This means that more than t_b bit-errors cannot be detected. Perfect codes can be seen as the most efficient codes, since they use the complete message space. A drawback of using perfect codes is that all received messages, including the messages containing errors will decode to a solution. More than t_b-bit errors will not be detected [25].

Gilbert-Varsharmov Bound

The Gilbert bound: $\sum_{d=0}^{2t_b} \frac{n_b!}{d!(n_b-d)!} \leq 2^{(n_b-k_b)}$, gives the smallest size of the message space (2^{n_b}) that guarantees that there will be a t_b-error correcting code for k_b data bits. Most error codes are chosen between the Hamming and GV bounds. This means that often, even though the capacity to correct errors may have been exceeded, the code can still detect unrecoverable errors [25].

2-3-3 Security Measures

To measure the state of the current and improved security of a system, several measures can be used. First of all there is the brute force attack. In addition, this section discusses several probabilities relating to the occurrence of errors and success of an attack.

Starting with some notation for the problem at hand, define

$$\mathbb{B} = \{0, 1\} \tag{2-3.4}$$

$$l \in \mathbb{L} = \mathbb{B}^n \tag{2-3.5}$$

where $\mathbb{L}$ is the set that consists of all possible sequences l with a certain number of bits. Since AETC is applied, only one bit is required for the transmissions, that is 0 represents the $-$ sign and 1 represents the $+$ sign. The sequence itself represents if there is a sampling. Hence, the set $\mathbb{K}$ that consists of the meaning of the sequence can be represented as:

$$k_m \in \mathbb{K} = \{\text{`0'}, \text{`1'}, \text{dummy}, \text{empty}\} \tag{2-3.6}$$

k_m is the meaning of a certain sequence, "empty" means the sequence is meaningless. A mapping from the sequences to the meanings f can be defined as:

$$f : \mathbb{L} \to \mathbb{K}, f(l) \in \mathbb{K}\, l \in \mathbb{L} \tag{2-3.7}$$

Define $\mathbb{F}$, the set that consists of all the possible mappings f. Define $L_{'0'}$, $L_{'1'}$, L_d the sets that consist all the possible sequences l such that $f(l) =$ '0', $f(l) =$ '1' and $f(l) =$ dummy respectively. Let $|L_{'0'}|$, $|L_{'1'}|$ and $|L_d|$ represent the cardinality of these sets.

Brute force attack

A security measure of a system is the vulnerability to brute force attacks. The brute force attack means that an attacker tries out all the possible alternatives. Table 2-3 shows the time required for a microchip to try out all the possible keys at 10^6 decryptions/μs for different key lengths. It is assumed that on average a brute force attack finds the correct key in half the time necessary to try all alternatives [26].

Table 2-3: Average time required for exhaustive key search [1]

Key size (bits)	**Number of alternatives**	**Time required at 10^6 decryptions/μs**
32	$2^{32} = 4.3 \cdot 10^9$	2.15 ms
56	$2^{56} = 7.2 \cdot 10^{16}$	10 hours
128	$2^{128} = 3.4 \cdot 10^{38}$	$5.4 \cdot 10^{18}$ years
168	$2^{168} = 3.7 \cdot 10^{50}$	$5.9 \cdot 10^{30}$ years

There are different possibilities for an attack scenario. An attacker could only have a single computer at his disposal or be able to utilize distributed computing. In case a national government is the attacker, it is plausible that it has a supercomputer at its disposal or employ a distributed attack using a set of networked smart phones [2]. Table 2-4 displays for different types of computing power, the number of Million Instructions Per Second (MIPS) available. The worst case assumption is made that one instruction is necessary to try a single alternative [26].

Table 2-4: Computational Power Estimates [2]

Computational Power	**MIPS**
Single Intel Quad Core	1.36×10^4
Infected computers China	5.89×10^4
BOINC	5.63×10^9
K Computer	1.05×10^{10}
World iPhones and iPods	1.39×10^{11}
Personal Computers in US	2.23×10^{12}

In [27], this security measure is combined with a performance measure of the system itself. The mean square tracking error is being used to evaluate the dynamic performance of the

system. The motivation for adding the dynamic performance, is the fact that a system which is highly oscillatory has less room for error. When the system response does not contain many oscillations, the system has a better tolerance for perturbations caused by malicious attacks.

Probability of an attacker arbitrarily finding a useful message

Assume the attacker knows the length of the bit sequence of the useful and dummy messages. In case a random test sequence (S_T) is used by the attacker in an effort to find a message, the chance of finding a message is equal to the total number of messages divided by the total number of useful messages, that is $|L_{`0'}| + |L_{`1'}|$:

$$P(S_T = L_{`0'} \vee S_T = L_{`1'}) = \frac{|L_{`0'}| + |L_{`1'}|}{2^{n_b}} \tag{2-3.8}$$

$$P(S_T = L_{`0'}) = \frac{|L_{`0'}|}{2^{n_b}}, P(S_T = L_{`1'}) = \frac{|L_{`1'}|}{2^{n_b}} \tag{2-3.9}$$

Equations 2-3.8 and 2-3.9 show that a larger amount of useful messages will result in a larger probability for an attacker finding a useful message by arbitrarily generating a sequence with the same length. While a lower amount of useful messages decreases this probability, the key will have to be updated more frequently.

Probability of a message being flipped by noise

The probability that an n_b-length sequence has t_b bits flipped is [25]:

$$P(t_b, n_b) = p^{t_b}(1-p)^{(n_b - t_b)} \frac{n_b!}{t_b!(n_b - t_b)!} \tag{2-3.10}$$

where p is the probability for each bit to be flipped, which comes from the Bit Error Rate (BER). The bits in a communications channel may be flipped by noise and this noise is assumed to be white. Meaning that over the bandwidth of the channel, noise has constant power at all frequencies. The BER is dependent on the amount of energy used to signal each bit.

The probability that a number of bits equal to d_H are flipped, can occur in a number of ways. Hence, the probability that a certain number of bits are flipped and the rest of the bits remains error free, needs to be multiplied by these combinations. In this way formula 2-3.10 is derived.

Probability of an attacker changing a message into a different message type

The probability of a successful change can be found by using the Hypergeometric distribution [28]. This distribution is defined in the following manner: the probability distribution

of the hypergeometric random variable i, the number of successes in a random sample of size m, selected from N items of which k_a are labelled success and $(N - k_a)$ labelled to be a failure:

$$P(i; N, m, k) = \frac{\binom{k_a}{i}\binom{N-k_a}{m-i}}{\binom{N}{m}} \tag{2-3.11}$$

The probability of an n_b-bit message to turn into another message type by using t_a bit changes:

$$P(i; N, t_a, |\{d|d_H(m_x, m_y) \leq t_a\}|) = \frac{\binom{|\{d|d_H(m_x,m_y)\leq t_a\}|}{i}\binom{N-|\{d|d_H(m_x,m_y)\leq t_a\}|}{t_a-i}}{\binom{N}{t_a}} \tag{2-3.12}$$

Where $|\{d|d_H(m_x, m_y) \leq t_a\}|$ is the number of messages of type m_y, at Hamming distance t_a away from the original message m_x and m_y is different than m_x. N is the total number of points at Hamming distance t_a away from the original message and $i = 1$, since only 1 message would already mean a success for the attacker.

For the three different message types '0', '1' and dummy, this results in equations 2-3.14 to 2-3.19 for all the different probabilities.

$$P(1; N, t_a, |\{d|d_H(m_{`0'}, d) \leq t_a\}|) = \frac{\binom{|\{d|d_H(m_{`0'},d)\leq t_a\}|}{1}\binom{N-|\{d|d_H(m_{`0'},d)\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.13}$$

$$P(1; N, t_a, |\{d|d_H(m_{`0'}, m_{`1'}) \leq t_a\}|) = \frac{\binom{|\{d|d_H(m_{`0'},m_{`1'})\leq t_a\}|}{1}\binom{N-|\{d|d_H(m_{`0'},m_{`1'})\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.14}$$

$$P(1; N, t_a, |\{d|d_H(m_{`1'}, d) \leq t_a\}|) = \frac{\binom{|\{d|d_H(m_{`1'},d)\leq t_a\}|}{1}\binom{N-|\{d|d_H(m_{`1'},d)\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.15}$$

$$P(1; N, t_a, |\{d|d_H(m_{`1'}, m_{`0'}) \leq t_a\}|) = \frac{\binom{|\{d|d_H(m_{`1'},m_{`0'})\leq t_a\}|}{1}\binom{N-|\{d|d_H(m_{`1'},m_{`0'})\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.16}$$

$$P(1; N, t_a, |\{d|d_H(d, m_{`0'}) \leq t_a\}|) = \frac{\binom{|\{d|d_H(d,m_{`0'})\leq t_a\}|}{1}\binom{N-|\{d|d_H(d,m_{`0'})\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.17}$$

$$P(1; N, t_a, |\{d|d_H(d, m_{`1'}) \leq t_a\}|) = \frac{\binom{|\{d|d_H(d,m_{`1'})\leq t_a\}|}{1}\binom{N-|\{d|d_H(d,m_{`1'})\leq t_a\}|}{t_a-1}}{\binom{N}{t_a}} \tag{2-3.18}$$

$$\tag{2-3.19}$$

Chapter 3

Solution

As was mentioned in the introduction, to improve the security of the single bit communication between sensor and controller, two methods are applied: encrypting the feedback channel and adding dummy messages. It is obvious that with the same encryption algorithm, the more bits an encrypted message has, the better it will be secured. However, the networked based feedback channel always has limited bandwidth, which limits the message length. Thus there is a trade-off when designing the length of the message. Hence, it is necessary to investigate the minimum message length, while satisfying the security constraints. A new mechanism that can both encrypt the single bit communication and add dummies is presented in this chapter.

3-1 General Architecture

The total scheme of the solution is shown in figure 3-1. It shows the additional blocks necessary to employ the encryption and key updates. The first step is to create a key by using a Key Derivation Function (KDF), which is based on the HKDF scheme [29]. For this key derivation, Source Keying Material (SKM) and a salt (non-secret key) are used to extract a pseudo random sequence with a high amount of entropy. In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes a password or passphrase. The primary function of salts is to defend against dictionary attacks and pre-computed rainbow table attacks. For the SKM, a random number will be generated with the current time as its seed. The pseudo random sequence is called the Pseudo Random Key (PRK). The PRK and the desired key length (n) form the input to the Pseudo Random Function (PRF). The output of the PRF is the key, which is a sequence that is close to uniformly random. The key is updated following a specified cryptoperiod, using a Key Update Function (KUF). In this case a HASH function is used on the previous

key. The HASH is a one-way function, which means that previous keys are protected when a key is compromised. The key is used as a starting point for the encryption algorithm, which determines the locations (bit sequences) for all the message types. The encryption scheme is shown in more detail in figure 3-2.

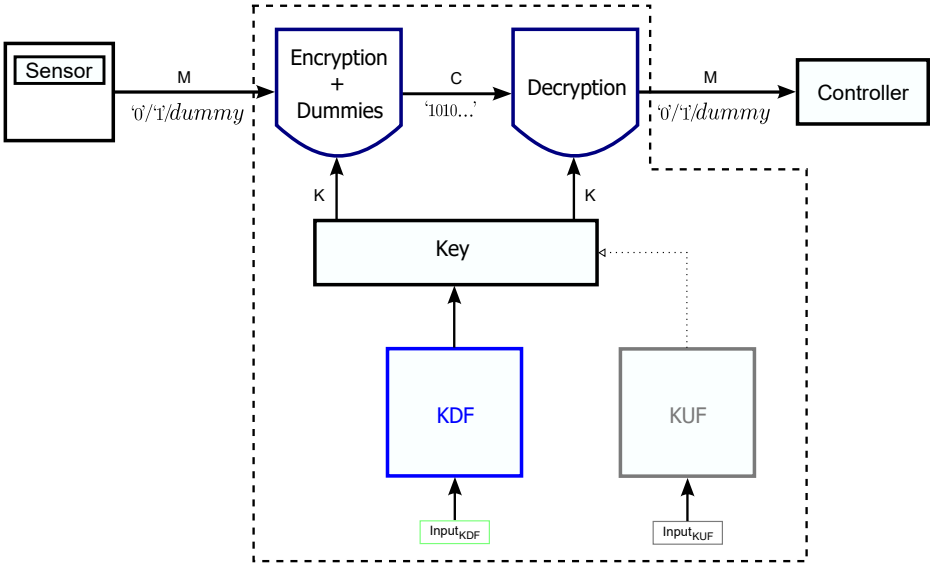

Figure 3-1: Block scheme of the solution

3-2 Mathematical Problem Formulation

3-2-1 Message Placement using Hamming Distance

Communication channels may be affected by noise or interference [23]. These disturbances could cause bits in the messages to be flipped. These flipped bits in turn, can lead to the misinterpretation of messages. When a message is misinterpreted by the controller, a wrong control action would be taken, which would destabilise the system instead of stabilising it. To prevent this described misinterpretation of messages, different messages should be distinguishable by placing them a certain distance apart from each other.

These messages can be placed by using different distance metrics. In [23], it is stated that the Hamming distance can be used when each bit has an equal probability of being flipped. This statement holds for wireless communication channels, where the message is transmitted on a bit-by-bit basis. Hence, for the mechanism at hand, it makes sense to use the Hamming distance.

By making sure the '0','1' and dummy messages are far from each other, i.e. have a large Hamming distance, the erroneous flipping of bits will not directly lead to a misinterpreta-

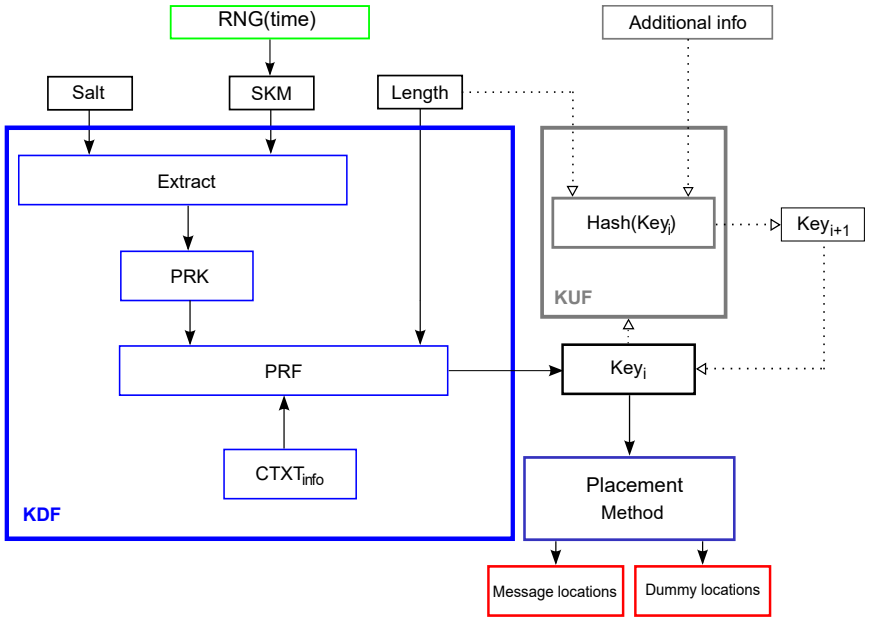

Figure 3-2: Detailed block scheme of the encryption

tion of the message. For instance, assume a bit sequence of 4 bits is used to send the single bit information through a wireless communication channel. The sequence '0000' could correspond to '0', '1111' corresponds to '1' and the bit sequences in between are positions which can be utilised for dummy messages. The hypercube plot of this example is shown graphically in figure 3-3; in which the dummies are depicted as red squares and placed at a Hamming distance of 2, away from the messages. The '0' message is represented by a downward blue triangle and the '1' message is represented by an upward green triangle

3-2-2 Optimal Message placement

An optimal function is presented to find the minimum bit length while satisfying several constraints.

$$\max_{f \in \mathbb{F}} |L_d| + |L_{`0'}| + |L_{`1'}| \tag{3-2.1}$$

$\mathbb{F}$ is the set of mappings from words to meanings, while words represent the encrypted one-bit transmissions and dummies; meanings represent the one-bit transmissions ('0', '1') and transmission of dummies (d). The locations of all the messages are defined in the

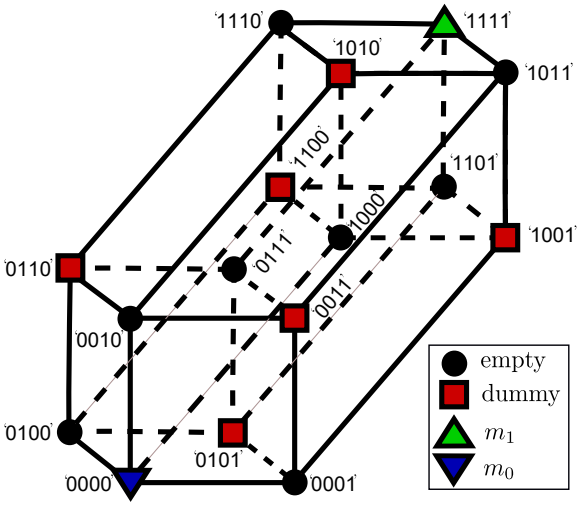

Figure 3-3: Possible message placement in a fourth order hypercube plot

following manner:

$$p_0 \in L_0 = \{f^{-1}(\text{`0'})\} \tag{3-2.2}$$
$$p_1 \in L_1 = \{f^{-1}(\text{`1'})\} \tag{3-2.3}$$
$$p_d \in L_d = \{f^{-1}(\text{dummy})\} \tag{3-2.4}$$

The constraints that have to be satisfied are:

- *Minimum number of messages and dummies:* A constraint on the minimum number of messages and dummies is necessary, because the same sequence cannot be transmitted more frequently than other sequences.

$$|L_{\text{`0'}}| \geq k_1 \tag{3-2.5}$$
$$|L_{\text{`1'}}| \geq k_1 \tag{3-2.6}$$
$$|L_d| \geq 2k_1 \tag{3-2.7}$$

- *Minimum Hamming distance between the messages and dummies and between the messages themselves:*

$$\widetilde{d}_H(m_{\text{`0'}}, d) = \min_{\substack{m_0 \in m_{\text{`0'}} \\ m_2 \in d}} d_H(m_0, m_2) \geq k_2 \tag{3-2.8}$$
$$\widetilde{d}_H(m_{\text{`1'}}, d) = \min_{\substack{m_1 \in m_{\text{`1'}} \\ m_2 \in d}} d_H(m_1, m_2) \geq k_2 \tag{3-2.9}$$
$$\widetilde{d}_H(m_{\text{`0'}}, m_{\text{`1'}}) = \min_{\substack{m_0 \in m_{\text{`0'}} \\ m_1 \in m_{\text{`1'}}}} d_H(m_0, m_1) \geq k_3 \tag{3-2.10}$$

In equations 3-2.8 and 3-2.9, the constant k_2 has to be larger or equal to 2. This is due to the fact that a Hamming distance equal to 0 would allow for locations to have multiple definitions, i.e. to be defined as a message '0' as well as a dummy. A Hamming distance equal to 1 would result in dummies and messages being located right next to each other and leave no room for bit flipping errors.
Constant k_3 in equation 3-2.10 has to be larger than the constant k_2, because it is most important not to confuse a '0' for a '1' than the other way around. If a '0' is interpreted incorrectly as a '1'; this would mean the controller would do the opposite control action as is required and accelerate the unstable behaviour.

- *Force new solution after key update:*
  Due to the possibility that a key update can result in the location of the encrypted initialisation message ($f_t(l_i)$) ending up at a location where the previous optimisation already allocated another encrypted '0' message; this could lead to the same optimisation solution. This means that the key update would result in the same key. To make sure that the key update results in a different pattern of dummies and messages, there should be a constraint that forces a different solution when the key is updated. This is achieved by appointing the location of an encrypted message or dummy of the previous optimisation to be empty in the current optimisation.

$$f_t(l_i) = empty, l_i \in f_{t-1}^{-1}(\text{'0'}) \cup f_{t-1}^{-1}(\text{'1'}) \cup f_{t-1}^{-1}(dummy \tag{3-2.11}$$

- *Percentage of messages:*

$$|L_d| + |L_{\text{'0'}}| + |L_{\text{'1'}}| < \rho 2^n \tag{3-2.12}$$

  The underlying idea of imposing this property is that by reducing ρ, the probability of an attacker randomly finding a sequence conveying information, i.e. a color different than **n**, is reduced.

3-2-3 Coloring Problem

The optimisation problem discussed in the previous section can also be seen as a coloring problem. Each of the four different location mappings can be represented by a different color. The solution space is a hypercube with 2^n vertices and each vertex must be assigned to one of these four colors. This leads to the following coloring problem formulation:
Consider a set of 4 possible colors $\mathcal{C} = \{\mathbf{1}, \mathbf{0}, \mathbf{d}, \mathbf{n}\}$ to be used to color a graph. The graph to be colored is the hypercube $\mathcal{H}_n = (\mathcal{V}, \mathcal{E})$, resulting from considering $\mathcal{V} = \mathbb{B}^n$, i.e. the set of n bit sequences, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V} = \{(v, v') | d_H(v, v') = 1\}$, with d_H the Hamming distance between sequences of bits.

Let us define the map $h : \mathcal{V} \to \mathcal{C}$ as the map providing the coloring of each of the vertices of the hypercube. And define the following four sets: $L_c = \{v \in \mathbb{B}^n \mid h(v) = c\}$, with $c \in \mathcal{C}$.

Problem 3-2.1. *Given a parameter set $\kappa = (k_1, k_2, k_3, \rho)$, color the graph $\mathcal{H}_n$ with colors in $\mathcal{C}$ so that the following properties are satisfied:*

1. $|L_{`0'}| = |L_{`1'}| = \frac{1}{2}|L_{\mathbf{d}}| = k_1$;
2. $d_H(v, v') \geq k_2,\ \forall\, v,\ v' \in L_j,\ j \in \{\mathbf{1}, \mathbf{0}, \mathbf{d}\}$;
3. $d_H(v, v') \geq k_3,\ \forall\, v \in L_i,\ v' \in L_j,\ i \neq j,\ i, j \in \{\mathbf{1}, \mathbf{0}, \mathbf{d}\}$;
4. $4k_1 \leq \rho 2^n$.

These conditions intuitively mean the following respectively:

1. there needs to be k_1 sequences associated to **1** and **0** and $2k_1$ sequences associated to **d**;
2. the hamming distance between sequences associated to the same color (except **n**) needs to be at least k_2;
3. the hamming distance between sequences associated to the different colors (except **n**) needs to be at least k_3;
4. the proportion of colors different than **n** to the total number of possible sequences needs to be smaller than ρ.

A consequence of using clusters will be that certain sequences and neighbours will appear with a higher frequency. This would give an attacker extra information about the location of useful messages, since all its neighbouring sequences are potential messages. When a message from the sensor is intercepted by the attacker, a neighbouring message can be inserted into the communication channel, with a high probability of being accepted by the controller.
The parameter k_3 controls the robustness to arbitrary bit flipping resulting in messages being misinterpreted as a different message; e.g. a '0' as a '1'.

Once a solution is found, one has access to $\frac{(|L_{`0'}|+|L_{`1'}|+|L_{\mathbf{d}}|)!}{|L_{`0'}|!\cdot|L_{`1'}|!\cdot|L_{\mathbf{d}}|!}$ unique solutions. This follows from the formula for distinguishable permutations [30]:

$$\binom{n}{n_1, n_2, ..., n_k} = \frac{n!}{n_1! n_2! ... n_k!}$$

Where n is the number of objects of which n_1 are of a certain kind, n_2 of another kind and n_k of the further kind, so that $n = n_1 + n_2 + ... + n_k$.
A different solution can be obtained by applying permutations on the order of the bits to the resulting solution, as reordering the bits does not alter the hamming distance between two sequences, i.e. $d_H(v, v') = d_H(r(v), r(v'))$, where r denotes a bit reordering. Since a random reordering could result in an equivalent solution i.e. dummies, and messages at the same locations there has to be an additional constraint which forces a different (unique) solution in case of a key update. The suggested solution follows from equation 3-2.11.

3-3 Minimum $\mathbf{n}$ using Coloring Problem Theories

The first question one may ask is what the minimum size of the hypercube is, that admits a solution, i.e.:

Problem 3-3.1. *Given a parameter set $\kappa = (k_1, k_2, k_3, \rho)$, find the minimum n_m, or an upper bound $\bar{n}_m$ for it, such that for every $n \geq \bar{n}_m \geq n_m$ a solution to Problem 3-2.1 exists.*

The Hamming bound gives a (partial) solution to Problem 3-3.1, because these bounds assume equidistant codes. Meaning that the Hamming distance between all messages is assumed to be equal.

3-3-1 Upper bound on the Minimum $\mathbf{n}$

It is well known that one can solve the 2-coloring problem on hypercubes efficiently (polynomial time). Solving such a problem on a hypercube $\mathcal{H}_m$, results in a partition of the graph vertices into two sets L_b, L_w such that $d_H(v, v') \geq 2$ for all vertices v, v' in the same set L_b or L_w. This fact will be used to construct a hypercube satisfying the necessary conditions. Employing this construction and additional tricks to enlarge the resulting distance between elements of one of the sets provides an upper bound on the minimum n necessary to solve the problem.

Consider a 2-colored hypercube $\mathcal{H}_m$, with $m > 2$, and the respective sets L_b and L_w. Let $k_d := \max\{k_2, k_3\}$. With $|L_b| = |L_w| = 2^{m-1}$. Now, let $p := 2^q > 1$, $r := 2^{m-1-q}$, with $q \leq m-1$, and partition L_b in r disjoint sets S_b^i, i.e. $S_b^i \cap S_b^j = \emptyset$, $\forall\, i \neq j$ and $\bigcup_{i=1}^{r} S_b^i = L_b$. By construction each S_b^i has cardinality p.

Definition 3-3.2 (Circular shift operator)**.** *Given a finite set S, the circular shift operator $\sharp_k : S^p \to S^p$, of order $k \in \mathbb{N}$, operating on sequences of length p of elements in S, is defined as:*

$$\sharp_k((s_1, s_2, s_3, \ldots, s_p)) := (s_{(1+k)\%k}, s_{(2+k)\%k}, \ldots, s_{(p+k)\%k})$$

where $a\%b := a - b\lfloor a/b \rfloor$.

Define for each set S_b^i, $i = 1, \ldots, r$, the sequences $\bar{s}_k^i \in (S_b^i)^p$, $k = 0, \ldots, p-1$ as:

$$\bar{s}_0^i := (s_1^i, s_2^i, \ldots, s_p^i), \tag{3-3.1}$$

$$\bar{s}_k^i := \sharp_k(\bar{s}_0), \tag{3-3.2}$$

where $s_j^i \in S_b^i$ for all $j = 1, \ldots, p$.

Given the fact that $d_H(s_j^i, s_l^k) \geq 2$ for any i, j, k, l such that $(i \neq k) \vee (i = k \wedge j \neq l)$, it automatically follows that $d_H(\bar{s}_j^i, \bar{s}_l^k) \geq 2p = 2^{q+1}$ for any i, j, k, l such that $(i \neq k) \vee (i =$

$k \wedge j \neq l$). Define now the set $\bar{S} \subset \mathbb{B}^{pm}$, as $\bar{S} = \bigcup_{i=1}^{r} \bigcup_{j=0}^{p-1} \bar{s}_j^i$. The cardinality of this set is $|\bar{S}| = 2^{m-1}$ by construction, and furthermore $d_H(\bar{s}, \bar{s}') \geq 2p$ for every pair of distinct elements $\bar{s}, \bar{s}' \in \bar{S}$, $\bar{s} \neq \bar{s}'$.

If now one lets $n \geq pm$ such that $2p = 2^{q+1} \geq k_d$, $2^{m-3} \geq k_1 \leq \rho 2^{pm-2}$, with $q \leq m-1$, a guaranteed solution to the problem can be found. Namely, by ordering the elements of $\bar{S}$ and marking the first k_1 elements as $\mathbf{0}$, the next k_1 elements as $\mathbf{1}$, the following $2k_1$ as $\mathbf{d}$ and the elements in $\mathbb{B}^n/\bar{S}$ as $\mathbf{n}$.

Thus, one can establish the upper bound $2^q m = \bar{n}_m \geq n_m$, where

$$m - 1 \geq q \geq \lceil \log_2 k_d \rceil - 1,$$

and

$$m \geq \max \left\{ \lceil 2^{-q} \left(\lceil \log_2 \left(\frac{k_1}{\rho} \right) \rceil + 2 \right) \rceil, \lceil \log_2 k_1 \rceil + 3 \right\}.$$

Which can be reduced to $\bar{n}_m = 2^{m-1} m$, with

$$m \geq \max \left\{ \lceil 2^{-m} 2 \left(\lceil \log_2 \left(\frac{k_1}{\rho} \right) \rceil + 2 \right) \rceil, \lceil \log_2 k_1 \rceil + 3, \lceil \log_2 k_d \rceil \right\}.$$

Such a bound provides a partial solution to Problem 3-3.1, and the construction described in this section a solution to Problem 3-2.1.

3-3-2 Placement Methods

Several solution methods for placing the messages are presented next. These methods do not assure a global optimum, however they can be used to find a feasible solution.

First Method

A possible method which can be used to enlarge the hamming distance between elements of $|L_{`0'}|$, $|L_{`1'}|$ and $L_{\mathbf{d}}$ is the following. Consider a set of vertices $S \subset \mathbb{B}^n$ of the hypercube $\mathcal{H}_n$ such that the distance between any two elements of that set $v, v' \in S$, $v \neq v'$ is $d_H(v, v') \geq \alpha$. Consider two arbitrary sequences $s_1, s_2 \in \mathbb{B}^r$ and denote by $\bar{s}_1, \bar{s}_2$ the complement of those sequences, i.e. the sequences with 1's where there were 0's and vice versa in the original sequence of bits. Such sequences satisfy the property that $d_H(\bar{s}_i, s_i) = r$.

Now, one can consider sequences in $\mathbb{B}^{n+2r}$ with the following coloring:

$$\begin{aligned} h((s, s_1, s_2)) &= \mathbf{1}, \\ h((s, \bar{s}_1, \bar{s}_2)) &= \mathbf{0}, \\ h((s, \bar{s}_1, s_2)) &= h((s, s_1, \bar{s}_2)) = \mathbf{d}, \end{aligned}$$

with $s \in S$, and any other sequence $s' \in \mathbb{B}^{n+2r}$ being colored by $\mathbf{n}$.

Such a construction would result in:

1. $|L_{`0'}| = |L_{`1'}| = \frac{1}{2}|L_{\mathbf{d}}|$;
2. $d_H(v, v') \geq \alpha$, $\forall\, v, v' \in L_j$, $j \in \{\mathbf{1}, \mathbf{0}, \mathbf{d}\}$;
3. $d_H(v, v') \geq 2r$, $\forall\, v \in L_{\mathbf{0}}$, $v' \in L_{\mathbf{1}}$,;
4. $d_H(v, v') \geq r$, $\forall\, v \in L_{\mathbf{d}}$, $v' \in L_j$, $j \in \{\mathbf{1}, \mathbf{0}\}$.

One can then combine this technique with the one from the previous section, by e.g. constructing the set S as the set $\bar{S}$ in the previous section. Then employing the trick in this section one can fine-tune the distance between the different sets of symbols.

Second method

In case $k_2 = k_3$, another method can be applied to enlarge the hamming distance between elements of $|L_{`0'}|$, $|L_{`1'}|$ and $L_{\mathbf{d}}$. This method considers a set of vertices $S \subset \mathbb{B}^n$ of the hypercube $\mathcal{H}_n$ such that the distance between any two elements of that set $v, v' \in S$, $v \neq v'$ is $d_H(v, v') \geq \alpha$. Consider three arbitrary sequences $s_1, s_2, s_3 \in \mathbb{B}^r$ and denote by $\bar{s}_1, \bar{s}_2, \bar{s}_3$ the complement of those sequences. Such sequences satisfy the property that $d_H(\bar{s}_i, s_i) = r$.

Now, one can consider sequences in $\mathbb{B}^{n+3r}$ with the following coloring:

$$\begin{aligned} h((s, s_1, s_2, s_3)) &= \mathbf{1}, \\ h((s, s_1, \bar{s}_2, \bar{s}_3)) &= \mathbf{0}, \\ h((s, \bar{s}_1, s_2, \bar{s}_3)) &= h((s, \bar{s}_1, \bar{s}_2, s_3)) = \mathbf{d}, \end{aligned}$$

with $s \in S$, and any other sequence $s' \in \mathbb{B}^{n+3r}$ being colored by $\mathbf{n}$.

Such a construction would result in:

1. $|L_{`0'}| = |L_{`1'}| = \frac{1}{2}|L_{\mathbf{d}}|$;
2. $d_H(v, v') \geq \alpha$, $\forall\, v, v' \in L_j$, $j \in \{\mathbf{1}, \mathbf{0}, \mathbf{d}\}$;
3. $d_H(v, v') \geq 2r$, $\forall\, v \in L_{\mathbf{0}}$, $v' \in L_{\mathbf{1}}$,;
4. $d_H(v, v') \geq 2r$, $\forall\, v \in L_{\mathbf{d}}$, $v' \in L_j$, $j \in \{\mathbf{1}, \mathbf{0}\}$.

This method uses one more sequence r compared to the previous method. However, this means a larger Hamming distances between message types can be achieved.

Complementary messages

When one of the two methods is used, an additional trick can be used. By also adding the complement of all the different message types to their respective groups a better coverage can be achieved. This trick can be applied when the method has not allocated these complements yet and the Hamming distance constraints remain satisfied. By complementing the sequences, the Hamming distance between the points themselves remains intact. However the minimum distance has become $\bar{s}$.For the first method this would mean adding:

$$\begin{aligned} h((\bar{s}, \bar{s_1}, \bar{s_2})) &= \mathbf{1}, \\ h((\bar{s}, s_1, s_2)) &= \mathbf{0}, \\ h((\bar{s}, s_1, \bar{s_2})) &= h((\bar{s}, \bar{s_1}, s_2)) = \mathbf{d}, \end{aligned}$$

For the second method this would mean adding the sequences:

$$\begin{aligned} h((\bar{s}, \bar{s_1}, \bar{s_2}, \bar{s_3})) &= \mathbf{1}, \\ h((\bar{s}, \bar{s_1}, s_2, s_3)) &= \mathbf{0}, \\ h((\bar{s}, s_1, \bar{s_2}, s_3)) &= h((\bar{s}, s_1, s_2, \bar{s_3})) = \mathbf{d}, \end{aligned}$$

These methods lead to two different placement algorithms using the cyclic shift.

Incremental Placement Algorithm

Another method which can be employed to find a feasible solution is; incrementally placing each consequent message. At each placement step the constraints have to be checked. The incremental placement method is shown in pseudo code below.

This technique can also be combined with the methods discussed in the previous section. This can be achieved by constructing the set S by first solving a reduced order problem with the Incremental Placement Algorithm. After which the distance between the different sets of symbols can be tuned.

3-3-3 Dummy Transmission

Dummies are sent when the difference between the current state $x_i(t)$ and the measured state $x_i(t_k)$ are equal to half of the threshold $\eta(t)$.

$$|x_i(t) - x_i(t_k)| = \frac{1}{2}\theta_i \eta(t) \tag{3-3.3}$$

Which results in sending a dummy alternated by sending a message. This also makes good use of the fact that twice the amount of dummies $|L_d|$ are available compared to $|L_{`0'}|$ and $|L_{`1'}|$. A drawback of transmitting dummies in this manner occurs when the time between consecutive messages becomes small. In this case, the communication channel will still be busy processing a dummy message and cause a delay for the next measurement, which is not able to come through. Hence this type of dummy transmission is feasible under the assumption of slow changing systems.

Algorithm 1 Random Incremental Placement Algorithm

procedure RNIPM
  while Random $p_1, f(p_1) \in \{dummy\}$ **do**

    while Solution set $S \neq \emptyset$

    Random $p_2 \in |p_2 - p_1| = k_1, f(p_2) \in \{m_{`0'}\}$

    Random $p_3 \in |p_3 - p_2| = k_1 \wedge |p_3 - p_1| \geq k_1, f(p_3) \in \{dummy\}$

    Random $p_4 \in |p_4 - p_3| = k_1 \wedge |p_4 - p_2| \geq k_1 \wedge |p_4 - p_1| \geq k_1, f(p_4) \in \{m_{`1'}\}$

    Random $p_5 \in |p_5 - p_4| = k_1 \wedge |p_5 - p_3| \geq k_1 \wedge |p_5 - p_2| \geq k_1 \wedge |p_5 - p_1| \geq k_1,$

    $f(p_5) \in \{dummy\}$

    etc.

  end while
end procedure

3-3-4 Key Update Strategy

When the cryptoperiod for the key is exceeded or there are no more messages available, a key update has to be performed. This key update creates a different mapping of messages and dummies The key is updated using a SHA-HASH function. SHA is chosen, because it is a cryptographic HASH function and it can generate key lengths of (limited) variable block sizes. This means the if a certain key length is sufficient, but not a multiple of the chosen block size, the HASH function will increase the key length. This occurs in the key update and key derivation step, since both steps use the same HASH function.

Chapter 4

Results

This chapter discusses the types of simulations that are going to be performed. First the simulations related to the minimum **n** are discussed, followed by the placement methods simulations.

4-1 Minimum n

The goal is to obtain a minimum **n** which is as small as possible. In the previous chapters, several tools have been discussed which can aid in choosing a suitable**n**. The cryptoperiod, bounds from source coding and the bound derived from graph theory.

4-1-1 Cryptoperiod

Assuming a brute force attack on the system, the cryptoperiod can be calculated using equation 2-2.8. This takes into account the improvements in computation power and the decay in security strength. Figure 4-1 shows the results when different types of brute force attacks are assumed. It shows that in case of an attack with a single Quad Core, 14 bits are sufficient to obtain a positive cryptoperiod. For the most extreme attack, the first positive cryptoperiod is reached at a key length of 41 bits. Which results in approximately 108 days (0.2961 year), after which the key has to be updated.
On average, the brute force attack succeeds in half the amount of time necessary to try all the options. In the model of [2], this was not accounted for. This means that with a key length of 41 bits, the key has to be updated in 54 days instead of 108 days.

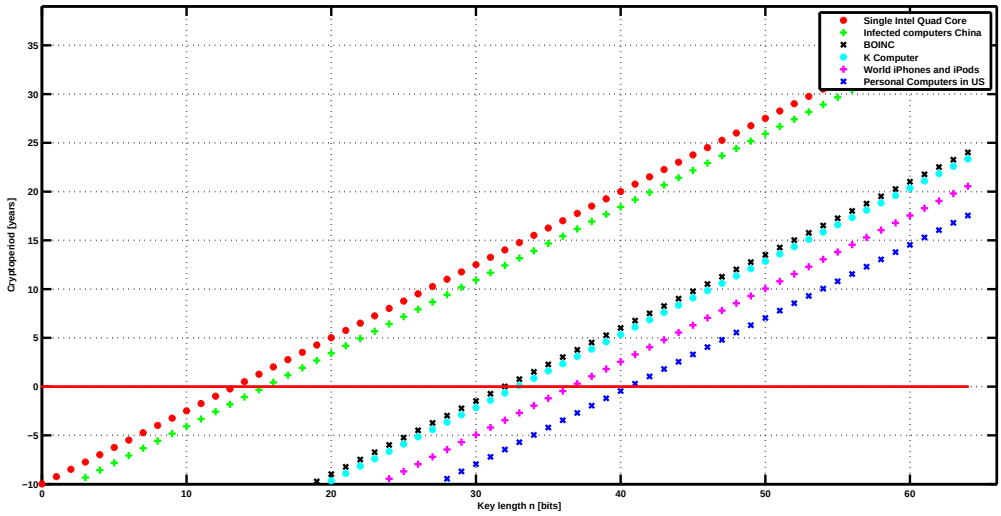

Figure 4-1: Cryptoperiod for different amounts of computation power

4-1-2 Case $k_2 = k_3$

In case it holds that the Hamming distance constraints k_2 and k_3 are equal; bounds from source coding can be applied to find bounds on the key length.

Bounds on the minimum $\mathbf{n}$

The Hamming bound gives a lower bound on the minimum $\mathbf{n}$ for the equidistant case. The VG bound gives an upper bound on the minimum $\mathbf{n}$. These are applicable for the equidistant case. The results are shown in figure 4-2. Both bounds are linearly dependent on error correction. For a specific cardinality (k_1) and number of bits that are allowed to be corrected, the minimum n has to be picked in between these bounds.

Upper bound on the minimum $\mathbf{n}$

The upper bound obtained from graph theory is shown in 4-3 with logarithmic scales for k_1 and k_2 It shows an unexpected result. Namely that increasing k_1 is more influential on n, than increasing the Hamming distance constraint k_2. However the coloring bound has a stronger dependency on k_1 than k_2. The coloring bound is compared to the Hamming bound in figure 4-4. This shows that the Hamming bound remains smaller or equal to the coloring bound. Hence the unexpected result is plausible, since the Hamming bound gives a minimum bound on $\mathbf{n}$ for equidistant codes.

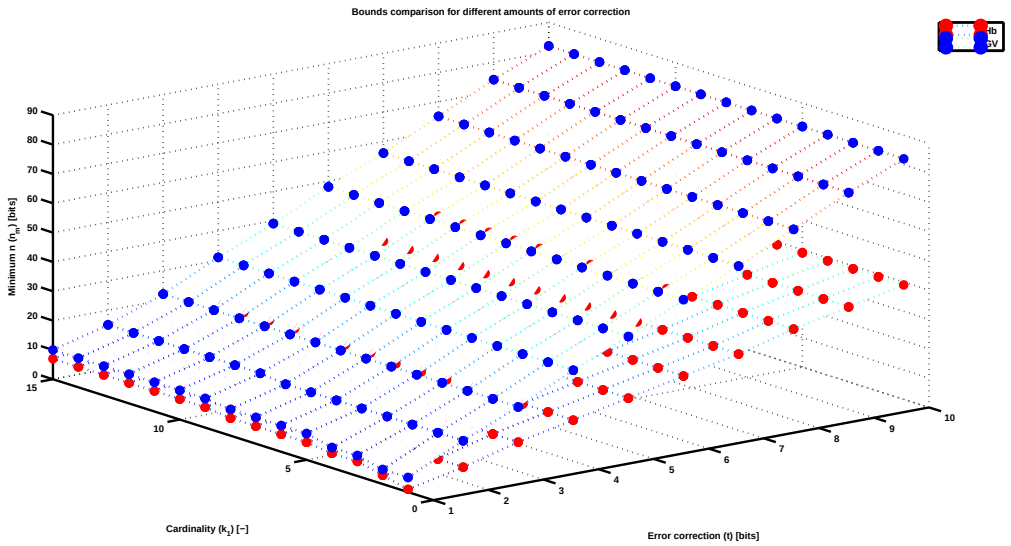

Figure 4-2: Comparison of bounds on minimum $\mathbf{n}$

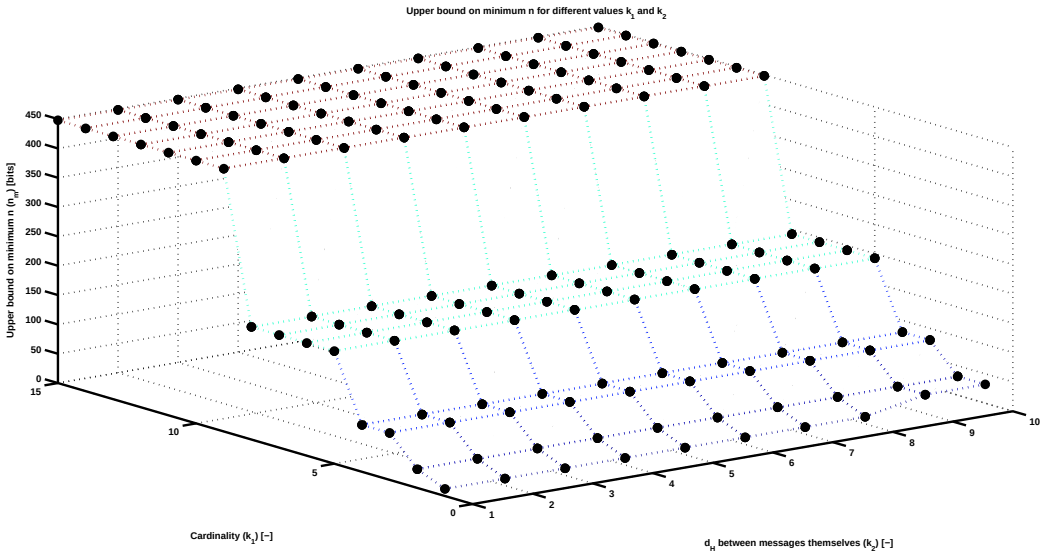

Figure 4-3: Upper bound on the minimum $\mathbf{n}$

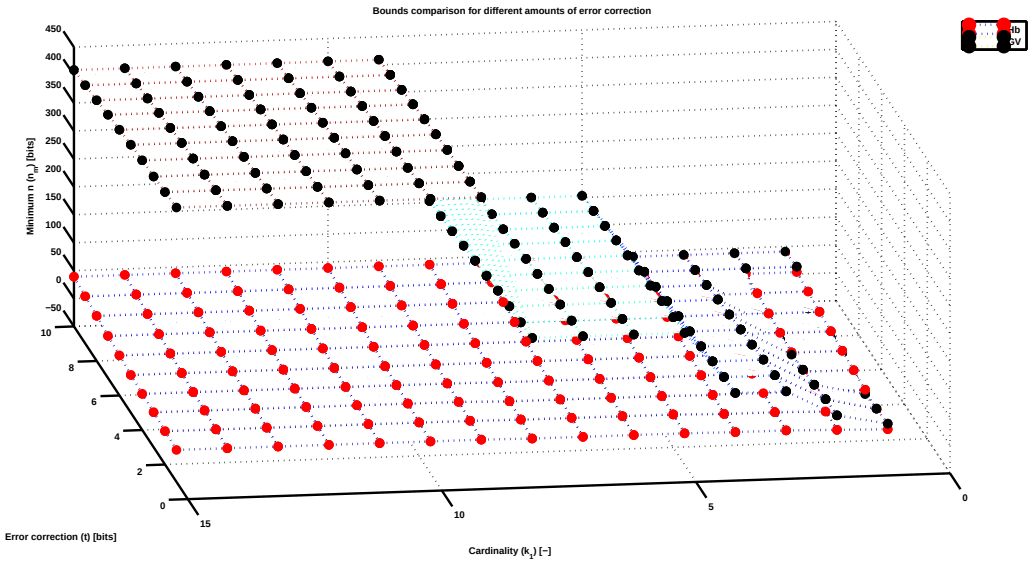

Figure 4-4: Comparison of coloring bound and Hamming bound on the minimum $\mathbf{n}$

In figure 4-5, the coloring and Hamming bound are plotted with the cryptoperiod constraint. It shows that the cryptoperiod constraint is equal or higher than the Hamming bound. This means that the minimum n can be chosen between the coloring bound and the cryptoperiod constraint.

4-2 Message Placement

The messages can be placed within the hypercube by using the different methods discussed in the previous chapter. First, the two placement methods following from graph coloring will be discussed. The third and final method is the Incremental placement method.

4-2-1 First Placement Method

The results of the first placement method are shown in figures 4-6 and 4-7. A total of eight and four messages are placed within the hypercube respectively. In the first case; two messages of m_0 and m_1 each are placed and 4 dummies. The second case places half the amount of each message type. This happens when the sequence s consists of only ones or zeroes. This causes the cyclic shift to result in the same sequence and both ways of creating dummies results in the same solution.

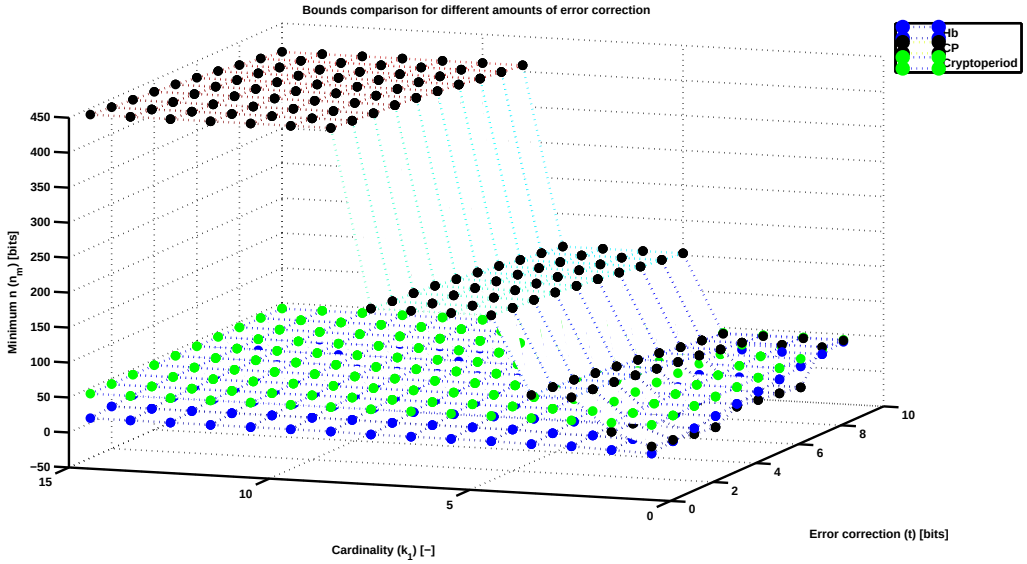

Figure 4-5: Comparison of coloring bound (black), Hamming bound (blue) and cryptoperiod constraint (green) on the minimum $\mathbf{n}$

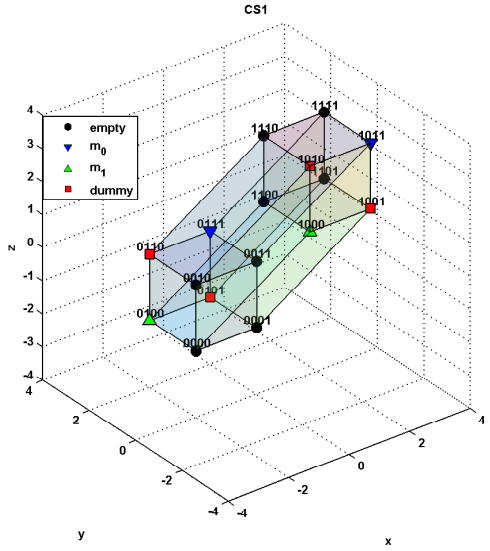

Figure 4-6: Message placement within fourth order hypercube using the first method

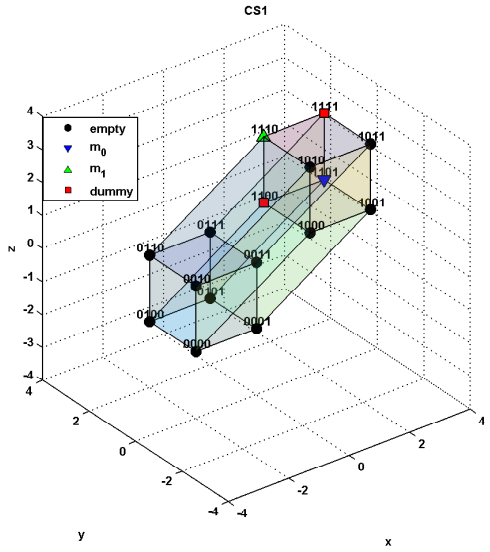

Figure 4-7: Message placement within fourth order hypercube using the first method

4-2-2 Second Placement Method

Figures 4-8 and 4-9 show the results of the second placement method. These results show 4 placed messages; a message of m_0 and m_1 each together with 2 dummies. Noticable is that the solution is always found in one of the two cubes. As a sidenote, permutations of these solutions can also be found using this method.

By also adding the complement of the messages to the solution, the available solution space can be used in a more efficient manner. Assuming that the Hamming distance constraints are still satisfied, since this means a minimum distance has now become equal to the length of s. The result of adding the complementary messages is shown in figure 4-10

4-2-3 Incremental message Placement

Figure 4-11 shows a result of the incremental message placement in a hypercube plot. In this case $k_1 = 1$ and $k_2 = k_3 = 2$. The dummies are located at '1100' and '1111' and are shown in green. A m_0 message (blue) is placed at '0101' and the m_1 message (red) is placed at '0011'.

4-2-4 Placement Method Comparison

When looking at the amount of messages, the cyclic shift placement methods have the best results. In this case, these methods are able to place double the amount of messages

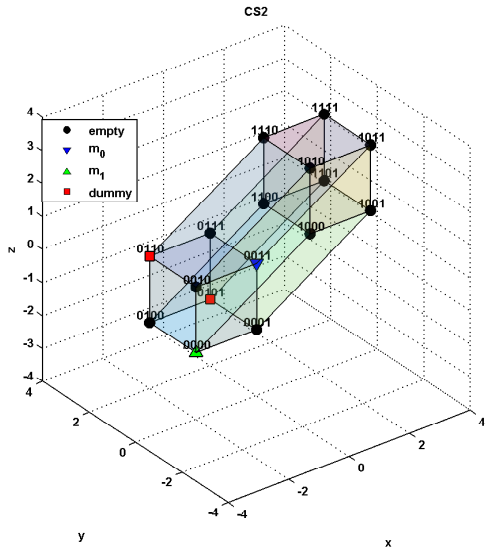

Figure 4-8: Message placement within fourth order hypercube using the second method

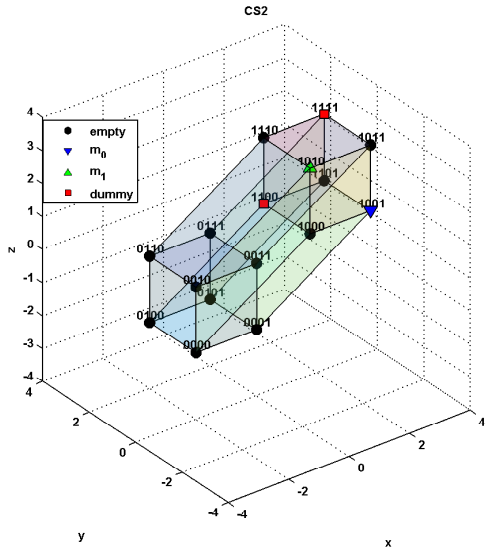

Figure 4-9: Different solution using the second method

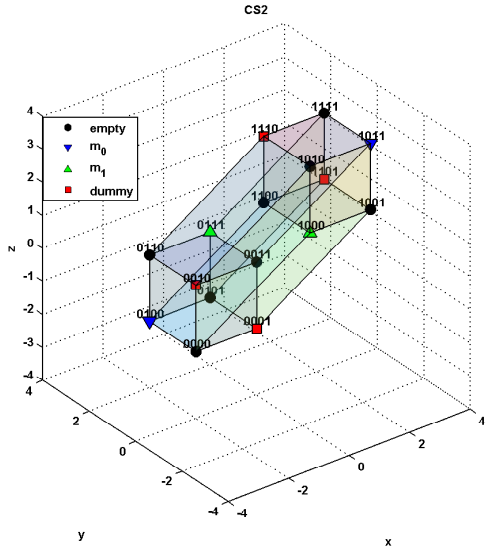

Figure 4-10: Message placement within fourth order hypercube using the second method with complementary messages

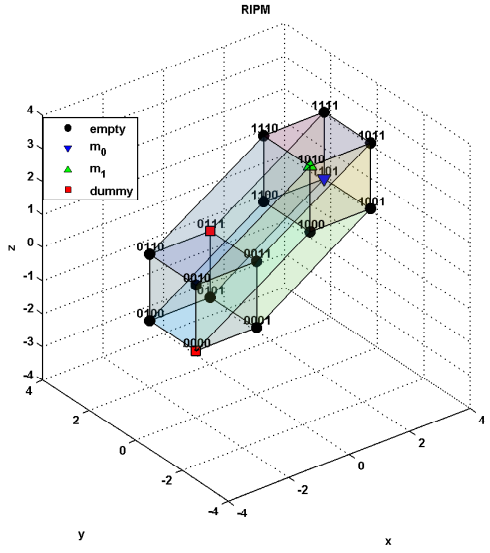

Figure 4-11: Incremental message placement $k_1 = 1$

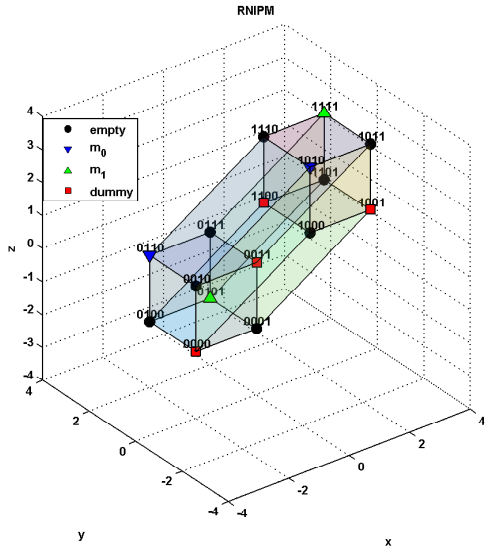

Figure 4-12: Incremental message placement $k_1 = 2$

compared to the incremental placement method. However, due to the random initialisation, this result is only attainable with the correct initial location for the first method. Otherwise the amount of placed messages will be less.
It is also noteworthy that the second method finds a solution with all locations close to one another, while the incremental locations are located far away from eachother. This could be explained by the fact that in the Incremental Placemtent method each point is generated at random.

The computation time necessary to place a message with different placement methods is compared in figure 4-13 for different key lengths. The computation time is the average of three runs. It shows that the cyclic shift methods perform faster than the incremental placement method. The first cyclic shift method is slightly faster than the second method. This difference is more noticable for smaller key lengths, which might be explained by the additional sequence introduced in the second cyclic shift method.

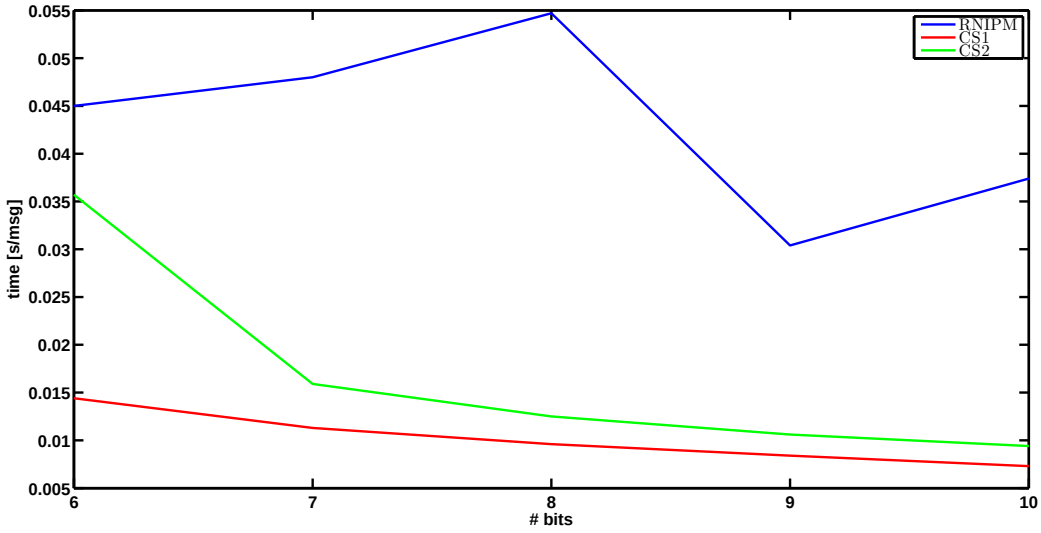

Figure 4-13: Comparison of the time required for the placement methods for different lengths

Chapter 5

Conclusion

5-1 Minimum n

The necessary key length can be designed using several different bounds. First of all, the bound following from the brute force attack can be utilised. In the most extreme attack scenario considered, the key should be at least 41 bits.
In case bit errors are also considered and $k_2 = k_3$, the minimum n can be designed satisfying the coloring upper bound and the Hamming bound. While always having 41 bits as an absolute lower bound. For the case $k_2! = k_3$, the cryptoperiod constraint and the graph theory bound remain valid and a minimum n can be picked between these bounds.

5-2 Placement Method

The first cyclic shift method is able to place messages in the fastest manner. Additionaly it is able to place more messages for a similar key length. From the importance of the time constraint, this method is chosen to perform the placement of messages. Another limitation of the random method is the fact that it is limited to a size of 12 bits. At this stage, the matrices created to check the Hamming between consecutive messages become too large. The cyclic shift methods do not suffer from this drawback and are able to handle up to 1024 bits. The only slight drawback of this method is a limitation on the the Hamming distance constraints. In the random method k_2 and k_3 can be chosen freely and in the cyclic shift methods this has to be done by increasing the length. Although it has to be noted that the random placement algorithm uses the maximum of k_2 and k_3 to place the next message. Hence, the suggested "freedom of choice" is also limited inthis instance.

5-3 Future Work

Suggestions for research directions for further investigation:

- Use a different HASH function for the Key derivation and update functions. NIST has recently updated its guideline of HASH functions to SHA-3. This HASH function is able to generate variable length hashes. This means it is not necessary to be have outputs as multiple of a certain block size e.g. 64, 128.
- Try to find the optimal message placement solution and compare it to the feasible solutions suggested in this thesis.
- Apply the cryptographic scheme to a laboratory setup of an NCS and test if the scheme is robust against attacks.

Bibliography

[1] J. van der Lubbe and Z. Erkin, "Sheets from the course: Security and cryptography."

[2] M. Juliato, C. Gebotys, *et al.*, "On the specification of symmetric key management parameters for secure space missions," in *Satellite Telecommunications (ESTEL), 2012 IEEE First AESS European Conference on*, pp. 1–6, IEEE, 2012.

[3] M. Mazo and M. Cao, "Asynchronous decentralized event-triggered control," *Automatica*, vol. 50, no. 12, pp. 3197–3203, 2014.

[4] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems.," in *HotSec*, 2008.

[5] N. Kottenstette, J. F. Hall, X. Koutsoukos, J. Sztipanovits, and P. Antsaklis, "Design of networked control systems using passivity," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 3, pp. 649–665, 2013.

[6] M. Abrams and J. Weiss, "Malicious control system cyber security attack case study–maroochy water services, australia," *McLean, VA: The MITRE Corporation*, 2008.

[7] J. Slay and M. Miller, *Lessons learned from the maroochy water breach*. Springer, 2008.

[8] A. Greenberg, "Hackers remotely kill a jeep on the highway–with me in it," *Wired, 21July*, 2015.

[9] M. Schöttle, "Hackers not crackers," *ATZelektronik worldwide*, vol. 10, no. 4, pp. 3–3, 2015.

[10] Y. Shoukry, J. Araujo, P. Tabuada, M. Srivastava, and K. H. Johansson, "Minimax control for cyber-physical systems under network packet scheduling attacks," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, pp. 93–100, ACM, 2013.

[11] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and secure networked control systems under denial-of-service attacks," in *Hybrid Systems: Computation and Control*, pp. 31–45, Springer, 2009.

[12] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, 2009.

[13] A. Cárdenas and R. Moreno, "Cyber-physical systems security for the smart grid," in *Cybersecurity in Cyber-Physical Systems Workshop*, 2012.

[14] M. Miskowicz, *Event-Based Control and Signal Processing*. CRC Press, 2015.

[15] E. D. Sontag, "Input to state stability: Basic concepts and results," in *Nonlinear and optimal control theory*, pp. 163–220, Springer, 2008.

[16] J. Van Leeuwen, *Handbook of theoretical computer science: Algorithms and complexity*, vol. 1. Elsevier, 1990.

[17] J. C. Van Der Lubbe and J. C. Lubbe, *Basic methods of cryptography*. Cambridge University Press, 1998.

[18] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management-part 1: General (revision 3), 2012," *NIST Special Publication*, pp. 800–57.

[19] A. Bogdanov, D. Khovratovich, and C. Rechberger, "Biclique cryptanalysis of the full aes," in *Advances in Cryptology–ASIACRYPT 2011*, pp. 344–371, Springer, 2011.

[20] S. Dziembowski, T. Kazana, and D. Wichs, "Key-evolution schemes resilient to space-bounded leakage," in *Advances in Cryptology–CRYPTO 2011*, pp. 335–353, Springer, 2011.

[21] A. Leier, C. Richter, W. Banzhaf, and H. Rauhe, "Cryptography with dna binary strands," *Biosystems*, vol. 57, no. 1, pp. 13–22, 2000.

[22] P. Sweeney, *Error control coding: from theory to practice*. John Wiley & Sons, Inc., 2002.

[23] W. Vereecken and M. Steyaert, *Ultra-wideband Pulse-based Radio: Reliable Communication Over a Wideband Channel*. Springer Science & Business Media, 2009.

[24] G. D. Forney, "Generalized minimum distance decoding," *Information Theory, IEEE Transactions on*, vol. 12, no. 2, pp. 125–131, 1966.

[25] A. Houghton, *Error coding for engineers*, vol. 641. Springer Science & Business Media, 2012.

[26] M. Whitman and H. Mattord, *Principles of information security*. Cengage Learning, 2011.

[27] W. Zeng and M.-Y. Chow, "Optimal tradeoff between performance and security in networked control systems based on coevolutionary algorithms," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 7, pp. 3016–3025, 2012.

[28] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and statistics for engineers and scientists*, vol. 5. Macmillan New York, 1993.

[29] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Advances in Cryptology–CRYPTO 2010*, pp. 631–648, Springer, 2010.

[30] K. Smith, *Nature of Mathematics*. Cengage Learning, 2011.

Glossary

List of Acronyms

NCS	Networked Control System
DoS	Denial-of-Service
AETC	Asynchronous Event-Triggered Control
KDF	Key Derivation Function
MIPS	Million Instructions Per Second
BER	Bit Error Rate

Nomenclature

$\eta(t_0)$	Initial value of the global threshold
$\lvert L_d \rvert$	Cardinality of the set of dummy messages
$\lvert L_{`0'} \rvert$	Cardinality of the set of '0' messages
$\lvert L_{`1'} \rvert$	Cardinality of the set of '1' messages
$\eta(t)$	Global threshold
$\eta_i(t)$	Local threshold for sensor i
$\mathbb{B}$	Binary set
$\mathbb{F}$	Set of all possible mappings
$\mathbb{K}$	Set of the possible mappings for each location
$\mathbb{L}$	Location set

$\mathbb{N}$	Natural numbers, including zero
$\mathbb{R}^+$	Positive real numbers
$\mathbb{R}_0^+ = \mathbb{R}^+ \cup 0$	Positive real numbers
μ	Predesigned parameter that determines how the threshold is updated
ρ	Design parameter
ρ	Percentage of messages in the total solution space
τ^c	Design parameter at which the sensors need to switch into listening mode
θ_i	Predesigned distributed parameter
$\underline{\alpha}$, $\overline{\alpha}$,α_v, α_e	Class $\mathcal{K}_\infty$ functions
ε	Measurement error
$\widetilde{d}_H$	Minimum Hamming distance
$\xi(t)$	State of the system
$\xi(t_0)$	Initial state of the system
$\{t_{r_i}^i\}$	Sequences of local sampling times for sensor i
c	Estimated computation power available in a specific year
c_e	Current computation power
d	Counter
$d_H(f_i, g_i)$	Hamming distance between elements f_i and g_i
$D_H(f_s, g_s)$	Hamming distance between strings f_s and g_s
E	Total amount of computation performed
f	Space of possible mappings into $\mathbb{F}$
f_i, g_i	String element i
f_s, g_s	Strings
g	Correction factor for the computational power gained
i	Random variable
k	Locally Lipschitz controller
k_1, k_2, k_3	Constants
k_a	Success
k_b	Number of messages
k_m	Meaning of a certain sequence
l	All possible sequences with a certain number of bits
l_i	Location i
$l_{m_{1,\text{`0'}}(t)}$	Location of the encrypted initialisation message (‘0’) during the current optimisation at time t
$l_{m_{1,\text{`0'}}(t-1)}$	Location of the encrypted initialisation message (‘0’) during the previous optimisation
m	Number of successes in a random sample
m_x, m_y	Message type (‘0’,1 or dummy), where m_y is different than m_x
N	Sample size

n	Key length
n_b	Bit sequence length
n_m	Minimum key length
n_s	Length of a string
p_0	Set of '0' message locations
p_1	Set of '1' message locations
p_d	Set of dummy message locations
T	Total amount of computation performed in t years
t_a	Number of bit changes made by an attacker
t_b	Number of bit errors
T_c	Cryptoperiod
t_c	Total computing time
$t^c_{r_c+1}$	Threshold update times
V	Continuously differentiable function
X	Hypergeometric random variable
'10..'	Bit sequences are depicted in between apostrophes
C	Ciphertext/Encrypted message
K	Key
M	Message
Salt	Non-secret key