Master Thesis
Industrial Engineering and Management
Smart Systems in Control and Automation

Secure and Private Formation Control of the Nexus Robot Using Fully Homomorphic Encryption

Author:
Kai Bellink
s2365294

Supervisors:
prof. dr. Bayu Jayawardhana
prof. dr. ir. Ming Cao

Abstract

Nowadays, Cyber-physical systems are developing in various industrial fields. Cyber-physical systems are physical systems that are connected through networks. Autonomous vehicles that communicate with each other are an example of these cyber-physical systems. The security of these cyber-physical systems is lacking and therefore, more research studies are focusing on the security of cyber-physical systems. The goal for this research is to implement a homomorphic encryption scheme in the formation control algorithm of the Nexus robot to increase the security and privacy. Homomorphic encryption schemes allow arithmetic operations to be performed on data that is encrypted. Different encryption schemes are explored that could be implemented to increase the security of the Nexus robot formation. The formation is simulated in RViz and the inter-agent distances together with the trajectory are monitored during various simulations. Based on these results the RMSE for the inter-agent distances is analyzed. The homomorphic encryption scheme is successfully implemented in the formation control algorithm and the performance of the formation control algorithm with the homomorphic encryption scheme are very similar to the basic formation control algorithm without the homomorphic encryption. Further research includes the implementation of the homomorphic encryption scheme in a more advanced formation control algorithm to improve the robustness and the implementation of the homomorphic encryption scheme in a real Nexus robot.

Contents

List of Figures

List of Tables

1 Introduction

Nowadays many physical systems are connected to computers and the internet through network communications. These combinations of a physical process (physical part), a computation (cyber part) and a communication (the link between the physical and cyber part) are called cyber-physical systems (CPS's) [1]. Cars that are connected to a wireless network, communicating with different devices inside and outside the vehicle is an example of such a CPS [2]. These ways of connecting physical systems with networks can result in additional benefits for the users. However these systems have also shown to be very vulnerable for malicious attacks, due to it's connectivity. Researchers have proven to be able to compromise cars by attacking the network [3]. Not only cars but also other CPS's and critical infrastructures such as water, transportation and electricity networks are subject to potential malicious attacks [4]. The security and privacy of the CPS's can be increased by the introduction of an encryption scheme. Encryption schemes are commonly used to combat eave-dropping attacks, which is one of the most basic attacks that requires low amount of resources. If an adversary is eave-dropping the system, the adversary is listening to the communication channel between the sensors, the controller and the actuator to acquire valuable data about the model and the controller, based on the transmitted data [5]. The encryption of data and information that has to be sent to or stored in the cloud becomes more important, since the concerns about potential risks involved with the loss of privacy, integrity and availability of the user data on cloud increases [6].
In this thesis an encryption scheme will be introduced to increase the security and privacy of a multi-agent formation control system. Formation control is a topic that is currently widely researched. Formations of multiple agents are used in applications that can not be performed by a single agent, such as surveillance, recognition, mapping, and rescue operations [7]. Other applications can be found in traffic related topics, where formation control is used in platooning. In platooning a string of vehicles travels together with small pre-defined inter-vehicle distances with a harmonized speed. Platooning has the potential to significantly increase safety and efficiency in traffic. Benefits of platooning are increased road throughput, traffic congestion mitigation and reduced energy consumption and exhaust emissions [8].
In this research a homomorphic encryption scheme will be implemented in a formation control algorithm to increase the security and privacy of the robot formation. Homomorphic encryption allows data to remain encrypted while arithmetic operations are performed on the encrypted data. Other encryption scheme require the data to be decrypted before arithmetic operations can be performed. Keeping the data encrypted will increase the safety of the system significantly.

1.1 System Description

The system that is considered for this research can be seen in Figure 1. The positions of the multi-agent formation are encrypted and then send to the cloud. The cloud is the controller where the arithmetic operations in the formation control algorithm are performed, while the data is encypted. The output of the controller will be the new velocities for the robot in the multi-agent formation system. The encrypted velocities are sent back to the robots where the data will be decrypted and processed. The operator that wants to move the formation sends his input variables to the cloud as well. The data transmission between the multi-agents formation and the cloud is now more secure and private due to the encryption of the data.

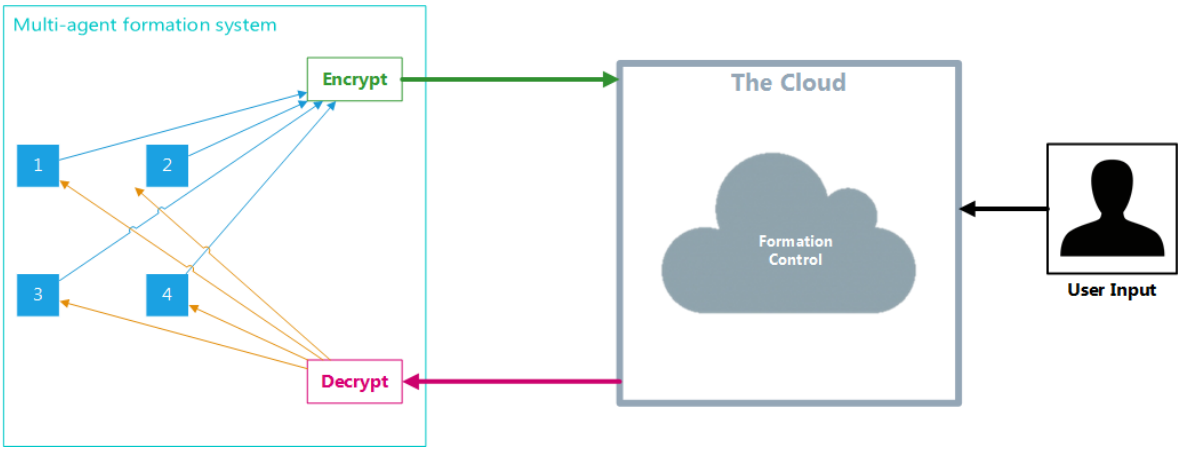


Figure 1: System description

1.2 Research Goal

The Nexus formation is currently lacking security and privacy. The communication with the robots in the formation is performed with the robot operating system (ROS). This operating system is widely used for the communication and control of different robotic components, however security features in this operating system are limited. The privacy and security will be increased by the implementation of a homomorphic encryption scheme. The goal of this thesis can therefore be formulated as:

"Implement a homomorphic encryption scheme in the Nexus robot formation control to increase security and privacy"

1.3 Research Questions

To eventually achieve the goal formulated in the section before and to give this research good guidance, a main research question is formulated. The main research question is:

"How to implement a homomorphic encryption scheme in the formation control algorithm for the Nexus robot?"

To answer this main research question the following sub questions are formulated:

1. *Which encryption scheme is suitable for the encryption of the Nexus formation?*
2. *Which formation control law is suitable for the multi-agent system?*
3. *What data must be encrypted?*
4. *How can the encrypted data be transmitted?*

When these sub questions are answered the main research question can be answered and the goal of thesis research can then be achieved.

1.4 Outline of Thesis

The outline of this thesis will be as follows. In the next chapter a literature study will be conducted to get a overview of the available encryption schemes and ways to control multi-agent formations. After this the homomorphic encryption scheme that will be used in

this thesis will be explained. Followed by a chapter that will explain the formation control algorithm. Then these two algorithms are combined and some simulations are performed, for which the simulation setup is first explained followed by the simulation results. After this the findings of this research will be concluded and discussed and an advise for further research will be given.

2 Literature review

Since the goal of this thesis is to implement a homomorphic encryption scheme in the formation control, a literature study is conducted on different encryption schemes and different formation control methods. This literature study will give a good overview of the available encryption schemes that can potentially be used as well as the different ways to control a formation of robots.

2.1 Encryption

In 1978 one of the first encryption schemes was introduced, the RSA encrytion scheme, named after the inventors Rivest, Shamir and Adleman [9]. This encryption scheme was followed by two semi-homomorphic encryption scheme, developed by El-Gamal in 1985 and Paillier in 1999 [10; 11]. El-Gamal developed a semi-homomorphic encryption scheme that allowed encrypted data to be multiplied. The other semi-homomorphic encryption scheme, developed by Paillier, allowed encrypted data to be added up. These encryption scheme are not fully homomorphic since they do not allow all arithmetic operations. Besides this, the amount of operations that can be performed on encrypted data with these semi-homomorphic encryption schemes is limited. In 2009 a fully homomorphic encryption scheme was developed by Gentry, which overcame the problem of limited amount of operations [12]. In general all the homomorphic encryption schemes can be categorized into three types of schemes, which are divided based on the kind of operations and the number of operations that can be performed on the encrypted data [13]. The first category is the *Partially Homomorpic Encryption* (PHE), which allows for one type of operations for an unlimited amount of times. The second category is the *Somewhat Homomorphic Encryption* (SWHE), which allows some types of operations for a limited amount of times. The third category is the *Fully Homomorphic Encryption* (FHE), which allows different types of operations to be performed unlimited amount of times [14].

In general a encryption scheme is called homomorphic if the following equation holds:

$$E(m_1) \star E(m_2) = E(m_1 \star m_2), \quad \forall m_1, m_2 \in M,$$

where E is the encryption algorithm, $\star$ is the arithmetic operation performed on the encrypted data and M is the set of all possible messages.

A homomorphic encryption scheme is mainly composed of the following functions: *KeyGen*, *Enc*, *Dec*. *KeyGen* is the function where the secret key and for some encryption schemes a second public key is generated for the encryption and decryption of the message m, often revert to as the plaintext. The *Enc* function is the encryption function in the homomorphic encryption scheme and transforms the message m into encrypted data which is commonly denoted by c and revert to as the ciphertext. *Dec* is the decryption function in the homomorphic encryption scheme and decrypts the ciphertext c to a plaintext format.

In Figure 2 the different homomorphic encryption schemes are summerized on a timeline until Gentry's first fully homomorphic encryption scheme. The encryption schemes that are most commonly used and revert to in literature are explained in more detail below.

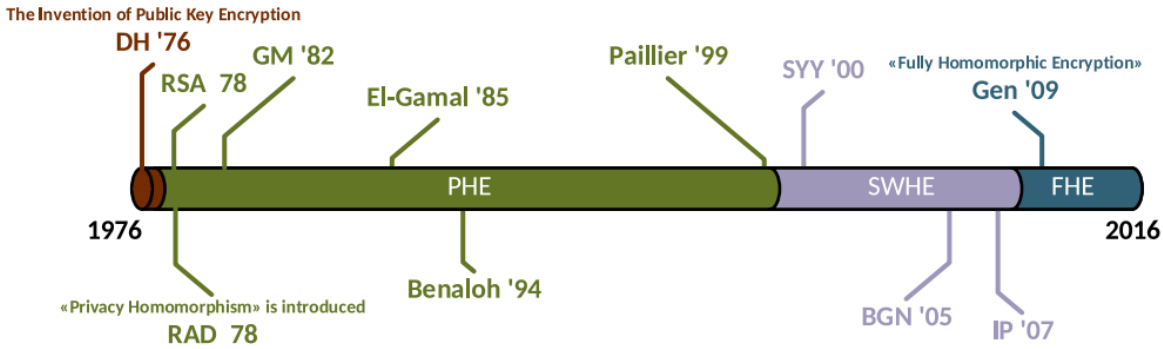


Figure 2: Timeline of different HE schemes until Gentry's first FHE scheme

Homomorphism in RSA

The RSA was one of the first PHE schemes and the description of the encryption scheme can be seen in Algorithm 1. The hardness of the factoring problem of the product of the two large prime numbers (p, q) determines the security of this encryption scheme [14]. The homomorphic property for two message m_1, $m_2 \in M$ is defined as:

$$\text{Enc}(m_1) * \text{Enc}(m_2) = (m_1^e \ (mod\ n)) * (m_2^e \ (mod\ n)) = (m_1 * m_2)^e \ (mod\ n) = \text{Enc}(m_1 * m_2).$$

Algorithm 1: RSA scheme

Key Generation KeyGen (p, q)
$Input : p, q \in P$
$Compute : n = p \cdot q, \phi(n) = (p-1) \cdot (q-1)$
$Choose : e \mid gcd(e, \phi(n)) = 1$
$Compute : d \mid (e \cdot d) \equiv 1 \ mod\ \phi(n)$
$Output : (P_k, S_k)$
$Public\ key : P_k = (e, n)$
$Secret\ key : S_k = d$
Encryption Enc (m, P_k)
$Input : m \in Z_n$
$Compute : c = m^e \ mod\ n$
$Output : c \in Z_n$
Decryption Dec (c, S_k)
$Input : c \in Z_{n^2}$
$Compute : m = c^d \ mod\ n$
$Output : m \in Z_n$

Homomorphism in ElGamal Encryption
The ElGamal encryption scheme is homomorphic over multiplication and the scheme can be seen in Algorithm 2. It uses a public key and is mostly used in hybrid encryption systems. The homomorphic property can be defined as:

$$\text{Enc}(m_1) * \text{Enc}(m_2) = (g^{x_1}, m_1 h^{x_1}) * (g^{x_2}, m_2 h^{x_2}) = (g^{x_1+x_2}, m_1 * m_2 h^{x_1+x_2}) = \text{Enc}(m_1 * m_2).$$

This shows that the ElGamal encryption scheme is homomorphic over multiplication. It does not support addition operations over the ciphertext.

Algorithm 2: ElGamal scheme

Key Generation KeyGen (n, g)
$Input : n, g$
$Compute : G = \{g, \cdots, g^n = 1\}$
$Choose : y \in Z_n$
$Compute : h = g^y$
$Output : (P_k, S_k)$
$Public\ key : P_k = (G, n, g, h)$
$Secret\ key : S_k = (x)$
Encryption Enc (m, P_k)
$Input : m \in Z_n$
$Choose : x \in \{1, 2, \cdots, n-1\}$
$Compute : c = (g^x, mh^x) = (g^x, mg^{xy}) = (c_1, c_2)$
$Output : c = (c_1, c_2)$
Decryption Dec (c, S_K)
$Input : c \in Z_{n^2}$
$Compute : s = {c_1}^y$
$Compute : m = c_2 \cdot s^{-1} = mg^{xy} \cdot g^{-xy}$
$Output : m \in Z_n$

Homomorphism in Paillier Cryptosystem
The encryption algorithm of Paillier, which can be seen in Algorithm 3, is homomorphic over addition. The algorithm computes the encryption of the messages m_1 and m_2 and it's application is manly used in electronic voting where all votes are encrypted and only the sum is decrypted [15]. The homomorphic property can be defined as:

$$\begin{aligned}\mathrm{Enc}(m_1) * \mathrm{Enc}(m_2) &= (g^{m_1} r_1^n \ (mod\ n^2)) * (g^{m_2} r_2^n \ (mod\ n^2)) \\ &= g^{m_1+m_2}(r_1 * r_2)^n \ (mod\ n^2) = \mathrm{Enc}(m_1 + m_2).\end{aligned}$$

This shows that the Paillier encryption scheme is homomorphic over addition.

Algorithm 3: Paillier scheme

Key Generation KeyGen (p, q)
$Input : p, q \in P$
$Compute : n = p \cdot q, \lambda = lcm(p-1, q-1)$
$Choose : g \in Z^*_{n^2} \mid gcd(L(g^\lambda \ mod\ n^2), n) = 1 \ where\ L(u) = (u-1)/n$
$Compute : \mu = L(g^\lambda \ mod\ n^2)^{-1}$
$Output : (P_k, S_k)$
$Public\ key : P_k = (n, q)$
$Secret\ key : S_k = (\lambda, \mu)$
Encryption Enc (m, P_k)
$Input : m \in Z_n$
$Choose : r \in z^*_n$
$Compute : c = g^m \cdot r^n \ mod\ n^2$
$Output : c \in Z_{n^2}$
Decryption Dec (c, S_K)
$Input : c \in Z_{n^2}$
$Compute : m = L(c^\lambda \ mod\ n^2) \cdot \mu \ mod\ n$
$Output : m \in Z_n$

Fully Homomorphic Encryption

The encryption scheme of Gentry is the first construction of a fully homomorphic encryption scheme. The encryption scheme starts with a some what homomorphic encryption scheme, which can be seen in Algorithm 4, that only supports a limited amount of operations. The problem with homomorphic encryption is that every ciphertext contains some noise, this is doubled when two ciphertexts are added up and even squared when two ciphertexts are multiplied. When this noise becomes larger than a certain threshold the ciphertext can no longer be decrypted. The growth of the noise limits the amount of operations that can be performed on encrypted data. Gentry overcame this problem with the introduction of *bootstrapping*. In this procedure an Evaluation function is used, which evaluates the encrypted data and return a new encryption with reduced noise. This refreshed ciphertext can then be used in new homomorphic operations.

Algorithm 4: Some what homomorphic encryption scheme

Key Generation KeyGen (R, B_i)
$Input : R, basis\ B_i$
$Compute : (B_{P_k}, B_{S_k}) = IdealGen\ (R, B_i)$
$Output : (P_k, S_k)$
$Public\ key : P_k = (R, B_i, B_{P_k}, S_{amp})$
$Secret\ key : S_k = B_{S_k}$
Encryption Enc (m, P_k)
$Input : m, P_k$
$Compute : e = S_{amp}(m, B_b i)$
$Compute : c = e\ mod\ B_{P_k}$
$Output : c \in \mathbb{R}\ mod\ B_{p_k}$
Decryption Dec (c, S_k)
$Input : c \in \mathbb{R}\ mod\ B_{p_k}$
$Compute : m = (c\ mod\ B_{S_k})\ mod\ B_i$
$Output : m$

2.2 Formation control

Formation control on multi-agent systems is a widely researched topic and therefore different formation control schemes have been developed. Throughout different studies the types of formation control and the different approaches are categorized in different types. In some literature studies the formation control schemes are categorized in three main types of formation control. The position-, displacement-, and distance-based control [16]. Other studies define three approaches for formation control: leader–follower based approach, virtual structure based approach and behavior based approach [17].

Position-based control: In position-based control the agents sense their own position with respect to a global coordinate system. The desired formation is controlled since every agent actively controls their own position. The desired formation is prescribed by the desired positions with respect to the global coordinate system. The following single-integrator modeled agents are considered, $\dot{p}_i = u_i$, where $p_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^n$ denote the position and the control input of agent i, with respect to the global coordinate system for $i = 1, \cdots, N$. The goal for the agents it is to move from an initial position to a desired position, while maintaining the formation shape. The desired position is known and denoted by $p^* \in \mathbb{R}^{nN}$, the objective is then to achieve $p \to p^*$, while satisfying $p_j - p_i = p_j^* - p_i^*$ for $i, j = 1, \cdots, N$ during the movement of the formation. The following control law can then be considered:

$$u_i = k_p(p_i^* - p_i), \tag{1}$$

where $k_p > 0$ is the gain.

Displacement-based control: In displacement-based control it is assumed that the agents sense relative positions of its neighboring agents with respect to the global coordinate system. This implies that the agents need to know the orientation of the global coordinate system, but the agents do not need any knowledge on the global coordinate system itself or their positions with respect to the coordinate system. The desired formation is controlled since every agent actively controls the displacements of their neighboring agents. The desired formation is specified by the desired displacements with respect to a global coordinate system. When a single-integrator modeled agent case is considered, similar to the position-based control $\dot{p}_i = u_i$, where $p_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^n$ denote the position and the control input of agent i, with respect to the global coordinate system for $i = 1, \cdots, N$. Besides this, it is assumed that the following relative positions are available to agent i, $p_{ji} := p_j - p_i$ for $j \in \mathcal{N}_i$ and $p^* \in \mathbb{R}^{nN}$ is known. Then the objective for the agents to satisfy becomes, $p_i - p_j = p_i^* - p_j^*$ for $i, j \in \mathcal{V}$, where $\mathcal{V}$ denotes the set of nodes in the graph. Using a consensus protocol the formation control problem can be solved considering the following control law:

$$u_i = k_p \sum_{j \in \mathcal{N}_i} w_{ij}(p_j - p_i - p_j^* + p_i^*), \tag{2}$$

where $k_p > 0$ is the gain and w_{ij} are real numbers associated with (i, j) for $i, j \in \mathcal{V}$. It is assumed that $w_{ij} > 0$ if $(i, j) \in \mathcal{E}$, where $\mathcal{E}$ denotes the edges. Let $e_p := p^* - p$ to obtain the following consensus dynamics:

$$\dot{e}_p = -k_p(L \otimes I_n)e_p,$$

where L is the Laplacian matrix and I_n denotes n-dimensional identity matrix.

Distance-based control: In distance-based control the agents are assumed to sense relative positions of their neighboring agents with respect to their own local coordinate systems. The orientation of local coordinate systems are not necessarily aligned with each other. The desired formation is controlled by actively controlling the inter-agent distances. The desired formation is prescribed by the desired inter-agent distances. If a directed formation is considered the single-integrator modeled agent formation control law can be defined as:

$$u_i = k_p \sum_{j \in \mathcal{N}_i} (\|p_j - p_i\| - \|p_j^* - p_i^*\|)(p_j - p_i), \tag{3}$$

where $\|x\|$ denotes the Euclidean norm of x.

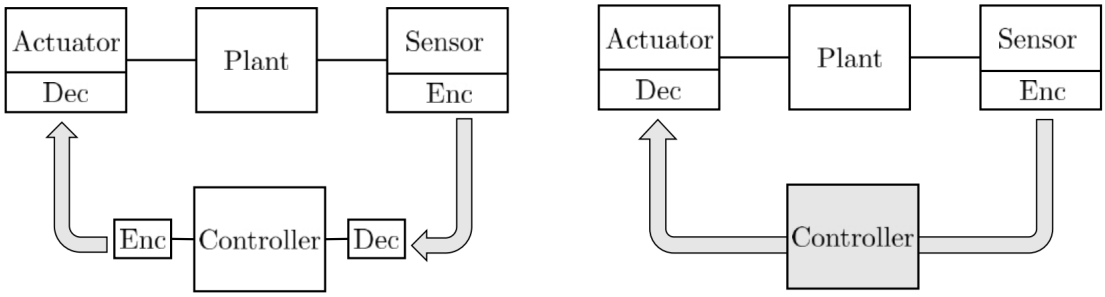
Table 1 shows for every type of formation control, the variables that are sensed, the variables that are controlled, the coordinate system that is used and the interaction topology. The coordinate system is either a local or a global coordinate system. The controlled variables are related to the interaction topology. For example, if the position of the robots is actively controlled, the robots can move to their desired position without interaction with other robots. In the case of distance-based control, where the inter-agent distances are controlled, the formation can be treated as a rigid body. The different robots in the formation have to communicate to maintain the desired shape of the formation.

	Position-based control	**Displacement-based control**	**Distance-based control**
Sensed variables:	Positions of agents	Relative positions of neighbors	Relative positions of neighbors
Controlled variables:	Positions of agents	Relative positions of neighbors	Inter-agent distances
Coordinate systems:	A global coordinate system	Orientation aligned local coordinate systems	Local coordinate systems
Interaction topology:	Usually not required	Connectedness or existence of a spanning tree	Rigidity or persistence

Table 1: The elements in position-, displacement- and distance-based formation control

3 Homomorphic encryption

The important feature of homomorphic encryption is that arithmetic operations can be performed while the data is encrypted. In Figure 3a the conventional encryption can be seen, where the data is encrypted before it is sent to the controller. When the encrypted data arrives at the controller it is decrypted to perform arithmetic operations. When the operations are performed the new data is again encrypted and sent back to the plant, where it is decrypted again to data that can be processed by the actuator. The drawback of this conventional way of encryption is that the secret key used for the encryption and decryption must be stored in the controller, which makes it less secure. Fully homomorphic encryption, which can be seen in Figure 3b, allows for arithmetic operations on encrypted data. The data is encrypted at the plant and sent to the controller, where the arithmetic operations are performed while the data remains encrypted. After the arithmetic operations the data is sent back to the plant where it is decrypted and processed. In this situation the secret key is only stored in the plant, which makes the system more secure than the conventional encryption.



(a) (b)

Figure 3: Conventional encryption and Fully Homomorphic Encryption (FHE)

3.1 Learning With Errors - LWE

In this chapter the LWE-based fully homomorphic encryption scheme, that is used in this thesis, will be explained. First the encryption and decryption steps in the encryption scheme will be explained.

The integer to be encrypted belongs to the set $[p]$ and it is denoted by $\mathcal{P}$, which is called the plaintext space. An element $m \in \mathcal{P}$ is called a message or a plaintext. The value of p can be a power of 10 such that $|m| < p/2$. $\mathbb{Z}_q$ is a set of integers modulo[1] q, where $q = Lp$, with L being a power of 10. Now in order to encrypt and decrypt a message or plaintext m a secret key is required. Let sk be an integer vector of size N such that $sk \in \mathbb{Z}_q^N$.

For every new message m that is encrypted a new random matrix A $\in \mathbb{Z}_q^{n \times N}$ is sampled, where n is the number of elements of the column vector $m \in \mathcal{P} = [p]^n$. Then a new vector e $\in [r]^n$ is randomly sampled where $r < L$, such that $|e_i| < L/2$ for i = 1,$\cdots$, n. With these components b can be computed by

$$b \leftarrow (-A \cdot sk + Lm + e) \; mod \, q. \tag{4}$$

[1] x mod q is the remainder after division of x by q. The remainder is supposed to be an element of $[q]$, if the remainder is greater than or equal to $q/2$, q is subtracted from the remainder to make it negative, e.g., 17 mod 10 = -3 $\in$ [10]

The ciphertext $\mathbf{m}$ of the plaintext m is obtained by $\mathbf{m} = [\mathrm{b}, \mathrm{A}] \in \mathcal{C} = \mathbb{Z}_q^{n\times(N+1)}$, where $\mathcal{C}$ is the ciphertext space. In order to decrypt the ciphertext $\mathbf{m}$, the secret key vector s must be defined as s $:= [1, sk^T]^T$. The ciphertext $\mathbf{m}$ is then decrypted as

$$\left\lceil \frac{(\mathbf{m} \cdot s)\ mod\ q}{L} \right\rfloor = \left\lceil \frac{Lm + e}{L} \right\rfloor \rightarrow m, \tag{5}$$

where $\lceil \cdot \rfloor$ is the rounding operation for vectors[2].

The important element in the LWE encryption scheme is the vector e, which is called the "error" and is intentionally injected in the ciphertext. This "error" is eliminated when the ciphertext is decrypted and makes this encryption scheme more secure than other encryption schemes. The importance of this "error" vector will be explained in the next section.

3.2 Error injection in LWE scheme

The security of an encrypted system is defined by how hard it is to find the secret key sk. An adversary that wants to attack the encrypted system can easily retrieve the ciphertext $\mathbf{m}$ by eavesdropping the communication line. In some cases the adversary can even obtain the plaintext m, for example when someone predicts that an email starts with "Dear". In this case the adversary has obtained the ciphertext $\mathbf{m}$ and the related plaintext m. Since $\mathbf{m} =$ [b, A] the adversary can easily obtain (-A $\cdot$ sk + e) and A by subtracting Lm from b in (4). This shows that if there is no "error" vector e, the secret key sk can be found by solving a linear equation. Finding or "learning" the secret key sk was proved to be extremely hard. This problem is called the learning-with-errors (LWE) problem. The following example was introduced by [18], where $\bar{s} = [\mathrm{s}_1,\mathrm{s}_2,\cdots,\mathrm{s}_4]^T = [3,-5,1,0]^T \in \mathbb{Z}_{100}^4$ and the following sequence of linear equations with errors:

$$\begin{aligned}
32\mathrm{s}_1 + 17\mathrm{s}_2 - 5\mathrm{s}_3 + 8\mathrm{s}_4 + \mathrm{e}_1 &= 6 + \mathrm{e}_1 = 7 && \pmod{100}\\
-6\mathrm{s}_1 + 17\mathrm{s}_2 - 1\mathrm{s}_3 + 18\mathrm{s}_4 + \mathrm{e}_2 &= -2 + \mathrm{e}_2 = -3 && \pmod{100}\\
44\mathrm{s}_1 + 32\mathrm{s}_2 + 12\mathrm{s}_3 + 28\mathrm{s}_4 + \mathrm{e}_3 &= -16 + \mathrm{e}_3 = -15 && \pmod{100}\\
&\vdots
\end{aligned}$$

The error of 1,-1,1,$\cdots$ that is used in this example makes it harder to find $\bar{s}$. The error vector does not have to be large, a small error is enough. The LWE problem based cryptosystems are as much secure as it takes even the quantum computer a long time to solve.

3.3 Security level of LWE encryption scheme

The security level of the LWE encryption scheme is determined by the parameters used in the encryption and decryption of the plaintext. The security of the encryption scheme that follows, based on the chosen parameters, is expressed as λ-bit secure. This level of security can be explained with a game between a challenger and an adversary, in which the adversary submits two messages m_1 and m_2 to the challenger. The challenger than chooses one of

[2]Negative numbers are rounded away from zero, e.g., $\lceil$-2.5$\rfloor$ = -3 while $\lceil$-2.4$\rfloor$ = -2. $\lceil a \rfloor$ implies component-wise rounding when a is a vector.

the two messages and returns it to the adversary after he has encrypted the message. The adversary then tries to guess which of the two message he has received from the challenger, by looking at the ciphertext. If the ratio of correct guesses by the adversary is not meaningfully greater than 0.5 after several attempts, the encryption scheme is said to be indistinguishable. This game can be expressed as,

$$(\text{Computation complexity of } \mathcal{A}) \times \left(\frac{1}{|0.5 - \text{Success Probability of } \mathcal{A}|} \right) > 2^{\lambda}, \tag{6}$$

where $\mathcal{A}$ is the algorithm used by the adversary to predict the message. Based on this equation it becomes clear that a larger λ-bit security implies that the computational complexity of the adversary's algorithm $\mathcal{A}$ increases for a success rate that is not very different from 0.5.

Besides this way of representing the security of the LWE encryption scheme, an online tool can be used to estimate the security of the LWE encryption scheme. Based on the parameters p, L, r and N the 'LWE-estimator' computes the number of operations to attack the encryption scheme with various attack algorithms and gives the value for λ. The following parameters where used for an example:

$$p = 1e4; \quad L = 1e4; \quad r = 1e2; \quad N = 20;$$

This gave the following results,

$$\begin{aligned} \text{usvp: rop:} &= 2\hat{}29.7, \ldots \\ \text{dec: rop:} &= 2\hat{}32.3, \ldots \\ \text{dual: rop:} &= 2\hat{}31.3, \ldots \end{aligned}$$

The rop values show the number of operations required for every attack algorithm, *usvp*, *dec* and *dual*. Based on these results it can be concluded that the LWE encryption, with the chosen parameters, is at least 29.7-bit secure. Finally it can be said that the λ security level has roughly the following property:

$$\frac{N}{\log q - \log r} \propto \frac{\lambda}{\log \lambda}. \tag{7}$$

This shows that increasing the parameter N will also increase the level of security.

3.4 Homomorphic arithmetics

In this chapter the homomorphic property of the LWE-based encryption scheme and the homomorphic arithmetics are explained. The homomorphic arithmetics makes this encryption scheme very interesting for control engineering field. The homomorphic arithmetics mean that one can perform addition and multiplication operations on data while that data is encrypted. The homomorphic arithmetic for two messages m_1 and m_2 that are multiplied with each other can be defined as:

$$\text{Dec}(\text{Enc}(m_1) *_{\mathcal{C}} \text{Enc}(m_2)) = m_1 *_{\mathcal{P}} m_2, \tag{8}$$

where $*_{\mathcal{C}}$ and $*_{\mathcal{P}}$ are the binary operations in the ciphertext space $\mathcal{C}$ and the plaintext space $\mathcal{P}$. Where Dec and Enc are the decryption and encryption functions. The LWE-based encryption scheme allows for homomorphic addition and homomorphic multiplication. The homomorphic addition can be defined as:

$$\begin{aligned}\mathrm{Enc}(m_1) +_{\mathcal{C}} \mathrm{Enc}(m_2) &= \mathbf{m_1} + \mathbf{m_2} \\ &= [-\mathrm{A}_1 \cdot \mathrm{sk} + Lm_1 + \mathrm{e}_1, \mathrm{A}_1] + [-\mathrm{A}_2 \cdot \mathrm{sk} + Lm_2 + \mathrm{e}_2, \mathrm{A}_2] \\ &= [-(\mathrm{A}_1 + \mathrm{A}_2) \cdot \mathrm{sk} + L(m_1 + m_2) + (\mathrm{e}_1 + \mathrm{e}_2), (\mathrm{A}_1 + \mathrm{A}_2)].\end{aligned} \tag{9}$$

The decryption of the ciphertext after the addition of the two encrypted messages can be defined as:

$$\mathrm{Dec}(\mathbf{m_1} + \mathbf{m_2}) = \left\lceil \frac{(\mathbf{m_1} + \mathbf{m_2}) \cdot s \bmod q}{L} \right\rfloor = \left\lceil \frac{L(m_1 + m_2) + \mathrm{e}_1 + \mathrm{e}_2}{L} \right\rfloor = m_1 + m_2, \tag{10}$$

under the condition that $m_1 + m_2 \in [p]$ and that each element of $|\mathrm{e}_1 + \mathrm{e}_2|$ is less than $L/2$. The homomorphic multiplication is a bit different compared to the definition given at the beginning of this section. When two scalar plaintexts $m_1 \in [p]$ and $m_2 \in [p]$ are considered, where m_1 is the multiplicand and m_2 is the multiplier, then the previous encryption function is used for the multiplicand m_1 but the encryption function is changed for the multiplier m_2 as:

$$\mathbf{M_2} = \mathrm{Enc2}(m_2) = m_2 R + \mathrm{Enc}(0_{\log q \cdot (N+1) \times 1}) \in \mathbb{Z}_q^{\log q \cdot (N+1) \times (N+1)}, \tag{11}$$

where $0_{\log q \cdot (N+1) \times 1}$ is the plaintext of zero vector in $[p]^{\log q \cdot (N+1) \times 1}$ and where R is defined as:

$$R := [10^0, 10^1, 10^2, \cdots, 10^{\log q - 1}]^T \otimes I_{N+1}, \tag{12}$$

where $\otimes$ denotes the Kronecker product and I_{N+1} is the identity matrix of size $N+1$. Any vector c in $\mathbb{Z}_q^{1 \times (N+1)}$ can be represented as $c = \sum_{i=0}^{\log q - 1} c_i \cdot 10^i$ by using the radix of 10, where every component of c_i is one of the single digit 0,1,2$\cdots$,9. The decomposition function D, that decomposes the argument by its string of digits, can then be defined as:

$$D(c) := [c_0, c_1, \cdots, c_{\log q - 1}]. \tag{13}$$

For example, where $q = 10^2$ and $N = 2$:

$$D([40, 35, -27]) = D([40, 35, 73]) = [0, 5, 3, 4, 3, 7],$$

because [40,35,73] = [0,5,3]$\cdot 10^0$ + [4,3,7]$\cdot 10^1$. From this is follow that $c = D(c)R$. The multiplication between the two ciphertexts $\mathbf{m_1}$ and $\mathbf{m_2}$ can be performed as:

$$\mathbf{M_2} \times_{\mathcal{C}} \mathbf{m_1} := D(\mathbf{m_1}) \cdot \mathbf{M_2} \quad \in \mathbb{Z}_q^{1 \times (N+1)}, \tag{14}$$

where $\mathbf{M_2} = \mathrm{Enc2}(m_2)$ and D is the decomposition function. To show the homomorphic property, it is first noted that, with secret key s:

$$\begin{aligned}(\mathbf{M_2} \times_{\mathcal{C}} \mathbf{m_1}) \cdot s = D(\mathbf{m_1}) \cdot \mathbf{M_2} \cdot s &= D(\mathbf{m_1}) \cdot (m_2 R + \mathbf{O}) \cdot s \\ &= m_2 \mathbf{m_2} \cdot s + D(\mathbf{m_1}) \cdot \mathrm{e}_{\mathbf{M_2}},\end{aligned} \tag{15}$$

where $\mathbf{O} := \mathrm{Enc}(0_{\log q \cdot (N+1) \times 1})$ is a matrix of which each row is an encryption of plaintext 0 and where $\mathrm{e}_{\mathbf{M_2}} \in \mathbb{Z}_q^{\log q \cdot (N+1) \times 1}$ is the error vector inside the ciphertext $\mathbf{O}$. The homomorphic property for the multiplication of two ciphertexts can the be described as:

$$\mathrm{Dec}(\mathbf{M_2} \times_{\mathcal{C}} \mathbf{m_1}) = \left\lceil \frac{D(\mathbf{m_1}) \cdot \mathbf{M_2} \cdot s \bmod q}{L} \right\rfloor = \left\lceil \frac{m_2(Lm_1 + \mathrm{e}_1) + D(\mathbf{m_1}) \cdot \mathrm{e}_{\mathbf{M_2}}}{L} \right\rfloor = m_2 m_1, \tag{16}$$

under the condition that $m_2 m_1 \in [p]$ and

$$\left| \frac{m_2 \mathrm{e}_1}{L} + \frac{D(\mathbf{m_1}) \cdot \mathrm{e}_{\mathbf{M_2}}}{L} \right| < \frac{1}{2}. \tag{17}$$

However, if a ciphertext $\mathbf{m_1} \in \mathbb{Z}_q^{1 \times (N+1)}$ is multiplied with a plaintext $\mathrm{m}_2 \in [p]$ the multiplication can simply be performed as $\mathbf{m_1} m_2 \pmod q$ since this is considered as a repeated homomorphic addition.

3.5 Error growth

In the previous sections of this chapter the homomorphic arithmetics and the homomorphic properties are explained. However, there are some limitations to the homomorphic arithmetics that have to be considered. In (4) it can be seen that every plaintext message that is encrypted has the error vector e introduced during the encryption of the ciphertext, where $e < r/2$. The limitation of the homomorphic arithmetics is that the error can grow in the ciphertext after numerous arithmetic operations are performed on the ciphertext. If the maximum size of the error is reached, $\mathrm{ErrorMax}(\mathbf{m})$, then $\mathrm{ErrorMax}(\mathbf{m_1} + \mathbf{m_2}) = \mathrm{ErrorMax}(\mathbf{m_1}) + \mathrm{ErrorMax}(\mathbf{m_2})$ and $\mathrm{ErrorMax}(\mathbf{m_1} m_2) = m_2 \mathrm{ErrorMax}(\mathbf{m_1})$ with plaintext m_2 as can been seen in (9). The multiplication of two ciphertexts may cause a further increase of the error which can be seen in (17). The $\mathrm{ErrorMax}(\mathbf{m}_{\mathrm{prod}})$ is the maximum error in the product of the multiplier $\mathbf{M_2}$ and the multiplicand $\mathbf{m_1}$, which can be written as $\mathrm{ErrorMax}(\mathbf{m}_{\mathrm{prod}}) = m_2 \mathrm{ErrorMax}(\mathbf{m}_1) + 9(r/2) \log q$. The first term $m_2 \mathrm{ErrorMax}(\mathbf{m}_1)$ is natural, but the amount of the second term $9(r/2) \log q$ may always add whenever the multiplication operation is performed. This shows that when the arithmetic operations are performed many times the error may grow up to the point that it damages the message in the ciphertext. This happens when the size of the error in the ciphertext grows larger than L/2, which results in a different message after decryption compared to message when is would be multiplied in the plaintext space, as can be seen in the following example:

$$\begin{aligned} &\mathbf{m} = \mathrm{Enc}(1) && \mathbf{m} = \mathrm{Enc}(1) \\ &\mathrm{Dec}(3 * \mathbf{m}) = 3 && \mathrm{Dec}(3000 * \mathbf{m}) = 2999 \end{aligned}$$

The example shows the difference between the multiplication of a ciphertext with a small plaintext scalar and the multiplication of a ciphertext and a large plaintext scalar. The multiplication of the small plaintext scalar returns the expected message after decryption. However the message in the multiplication of the large plaintext scalar has been affected by the error in the ciphertext and therefore differs from the outcome when the operation was performed without encryption in the plaintext space, which would return 3000 instead of 2999.

3.6 Scaling factor

Besides the limitations of the homomorphic encryption scheme explained in the previous section regarding numerous arithmetic operations, the LWE based encryption scheme has another limitation. As can be seen in (5) the message that is returned after decryption is a rounded value. This means that messages with values such as 1.2 and 3.8 are round off to 1 and 4 respectively. However it might be necessary to keep the digit numbers during the encryption and decryption, to maintain a desired precision. The messages can be scaled to ensure the desired precision, which means that a message m = [1.2, 5.3, 38.1] can be scaled by introducing the scaling fraction $S_G := 1/10$. The scaling can then be defined as $\bar{\mathrm{m}} := [\mathrm{m}/S_G]$ such that $\bar{\mathrm{m}} = [12, 53, 381]$. Based on the precision that is required the scaling fraction can be adjusted accordingly.

4 Formation control

In the literature study in chapter 2 different formation control algorithms are discussed. In this chapter the formation control algorithm that will be used is explained. The position-based formation control algorithm is chosen for this thesis, since the simulation setup makes it very easy to monitor the positions of the robots in the global coordinate system. These positions can easily be translated into inter-agent distances. However this formation control is a very simplified version, since the homomorphic properties of the LWE-based encryption scheme does not allow very difficult mathematical operations. The formation control can be described as a directed graph which can be seen in Figure 4, where node 1 is the leader robot and node 2 and 3 are the following robots which are influenced by the position of leader robot. The details of this formation control algorithm are explained in the next section.

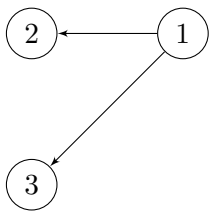


Figure 4: Graphical representation of the follower-leader formation control

4.1 Robot Model Description

The Nexus robots are used in this project for the formation control and the simulation. The Nexus robots are equipped with mecanum wheels, which allows the Nexus robots to move linearly along the x-axis and y-axis, rotate along the z-axis and all combinations of these movements. However, during the simulations it is assumed that the robot does not rotate around its z-axis and therefore always has the same orientation. The Nexus robot can then be modelled as a simple integrator:

$$\begin{bmatrix} \dot{x}_x \\ \dot{x}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_x \\ x_y \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_y, \tag{18}$$

where $u_x \in \mathbb{R}$ denotes the input velocity in the x-direction and $u_y \in \mathbb{R}$ denotes the input velocity in the y-direction.

4.2 Control Law

The different robots in the system send their coordinate position to the main controller. This controller determines the inter-distances of the following robots to the leading robot, and based on the desired distance that is set in the controller the velocity inputs for the following robots are determined. The dynamics of the following robots can be considered as single integrators:

$$\dot{A}_i = u_i \qquad \text{for i = 1,. . . ,k,}$$

where $A_i = [x_i(t), y_i(t)]^T \in \mathbb{R}^2$ is the position of the i-th robot [19]. The formation control for the following robots can then be written as:

$$\dot{A}_i = (A_1 - A_i) + D \qquad \text{for i = 2,. . . ,k,} \tag{19}$$

where $D = [d_x, d_y]^T$ is the desired relative distance between robot 1 and the i-th following robot. It can easily be checked that the i-th following robot and the leading robot 1 form a stationary formation on the plane when $A_i = A_1 + D$. This formation control algorithm can be combined with the following commonly used control law:

$$u_i = k_p(p_i^* - p_i), \tag{20}$$

where a gain $k_p > 0$ is introduced, p_i^* is the desired position of the i-th robot and p_i is the actual position of the robot [16]. Since the formation control algorithm is based on a leader-follower principal, the desired position of the following robots can be formulated as:

$$p_i^* = A_1 + D \qquad \text{for i = 2,. . . ,k,} \tag{21}$$

where $A_1 = [x_1(t), y_1(t)]^T \in \mathbb{R}^2$ is the position of the leading robot and $D = [d_x, d_y]^T \in \mathbb{R}^2$ is the desired distance between the i-th following and the leading robot. This results in the following control algorithm:

$$u_i = k_p((A_1 - A_i) + D) \qquad \text{for i = 2,. . . ,k.} \tag{22}$$

The k_p value in equation (22) might have a value of 0.6 which results in the best behavior of formation. However, if this same behavior must be obtained with encrypted data, the k_p value must also be encrypted. The encryption of a 0.6 value will result in a k_p value of 1 due to the rounding. For a k_p value of 0.6 to work the Enc2() function must be used as explained in chapter 3. In this case the Enc2() function will not work since the difference in position is also an homomorphic operation, which is then combined with the Enc2(0.6) function. To overcome this problem the k_p value is first scaled and then multiplied with the encrypted data. This works since a multiplication of a ciphertext with a plaintext integer is considered as numerous addition operations. The ciphertext that must be decrypted is then scaled down with a power of ten higher than the encrypted data. Consider the case where the following equation must be performed under encryption, $2.2 \times 0.6 = 1.32$. The value 2.2 and the k_p value 0.6 are both scaled and the scaled value of 2.2 is then encrypted, $\text{Enc}(2.2/S_G) \times (0.6/S_G) = \mathbf{m}$, where $S_G = 1/10$ is the scaling factor. After the decryption of the ciphertext $\mathbf{m}$, the message must be scaled down again. To obtain the desired value of 1.32, the ciphertext $\mathbf{m}$ is, after decryption, scaled down with the product of the two scaling factors used during encryption. This results in $\text{Dec}(\mathbf{m}) \times (S_G \times S_G) = 132 \times 1/100 = 1.32$. With this technique the control law proposed in equation (22) can still be used with encrypted data and any arbitrary value for k_p.

4.3 User-identification

Encryption schemes are very useful to protect information in such a way that others are not able to retrieve the original data. However, the encryption scheme in the nexus formation control only prevents hackers from obtaining the original control data and makes it therefore impossible to actively control the formation. Hackers can still send random data to the robot, as long as the data format has the same dimensions. Distorting the formations trajectory by sending random data to the robots may be enough for hackers to cause much damage. To prevent hackers from distorting the formations trajectory, an identification check is implemented in the algorithm. On the nexus robot a security number is encrypted and sent

with the other data that is needed for the formation control to the controller. When the data returns from the controller back to the nexus robot, the security number is first decrypted and compared to the original values. If these values match, the other input variables for the different robots are processed. When the original security number does not match the received and decrypted security number, the formation is stopped and a warning message is published, since this could mean that the data that is received does not come from a trusted party or the controller. With this new identification check, a hacker first has to know what the security number is before the trajectory can be distorted by sending malicious data to the robots. The addition of the security number to the encrypted scheme can be described as:

$$\mathbf{m} = \mathrm{Enc}(m, \sigma), \tag{23}$$

where $\mathbf{m}$ is the ciphertext of the plaintext m and $\sigma \in \mathbb{Z}$ is the security number, which is randomly generated for every new message that is encrypted.

5 Simulation setup

In this chapter the setup for the simulations is discussed. The first section of this chapter explains the robot operating system (ROS), that is used for the communication with the robots. After this the simulation software RViz is discussed that is used to simulate the formation. Finally the scripts that are used to let the formation drive with encrypted data are explained.

5.1 Nexus robots

The simulations are performed with Nexus robots that are used in the DTPA-lab. These robots are simulated in the simulation program RViz. The Nexus robot is equipped with omniwheels which allows the Nexus robot to move linearly along the x-axis and y-axis, rotate along the z-axis and all combinations of these movements. The movements that are considered during the simulations are all the linear movements along the x-axis and y-axis. The rotational movements are not considered, and with the linear movements the robot can still reach any destination.

Figure 5: The Nexus robot with mecanum wheels

5.2 ROS - Robot Operating System

The communication and controlling of the robots works via ROS, the robot operating system. ROS is an open source software framework for robot software development and controlling robotic components.

Nodes

The ROS system is a computational graph in which different nodes communicate with each other, see Figure 6. A node performs a computational process and can send messages to other nodes. The different nodes can perform different tasks, e.g. a node can control the robot's wheels, perform path planning or control the camera. The different nodes are organized in a publish and subscribe structure, which means that a node can publish to another node to send its message and a node can subscribe to another node to receive the message from that node [20].

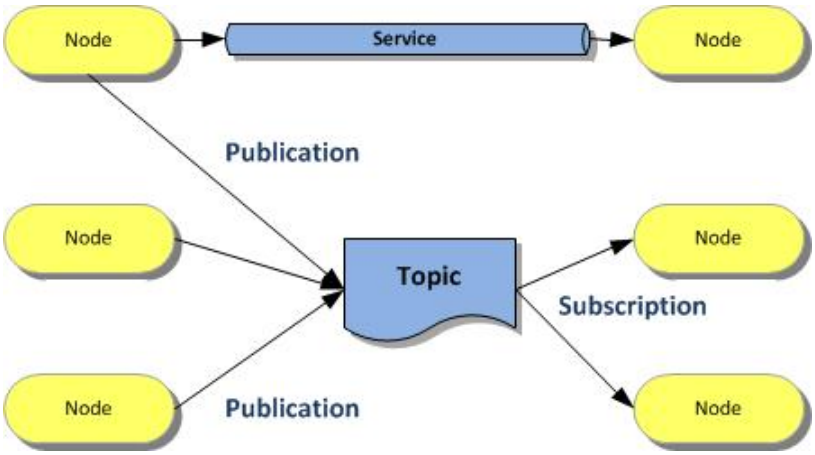


Figure 6: Graphical representation of ROS

Messages
The messages that are sent to the different nodes are simple data structures, consisting of typed fields. ROS has defined various types of messages that are supported. These messages vary from standard primitive types, such as integers, floating points and Booleans, to messages such as arrays of primitive types. Furthermore, ROS has also defined some more detailed messages, such as Geometry_msgs, Nav_msgs and Sensor_msgs. These types of messages contain very specific data, e.g. data related to the velocity of a robot or the position in a coordinate system [21].

Topics
The messages that are sent to other nodes are exchanged over topics, which are named channels. One or more nodes can publish to a topic and one or more nodes can subscribe or read the message on that topic. The topics are typed, which means that the type of message they can send to other nodes is set. The type to which the topics can be set are different, for example; numbers cannot be sent to another node if the topic uses a string type message [22].

ROS Master
The ROS master handles the publish and subscribe structure between the nodes and topics. The ROS master tracks the publishers and the subscribers to topics and services and notifies the different nodes of each other's existence. A camera node wants to publish an image on the topic "images", which is noticed by the master, Figure 7a. Another node wants to display images and therefore subscribes to the topic name "images" to display an image, if one is published to this topic, Figure 7b. The ROS master has now a publisher and a subscriber for the topic "images" and notifies both nodes of their existence. The camera node can now start transferring its data to the subscribed node that wants to display the image, Figure 7c [23].

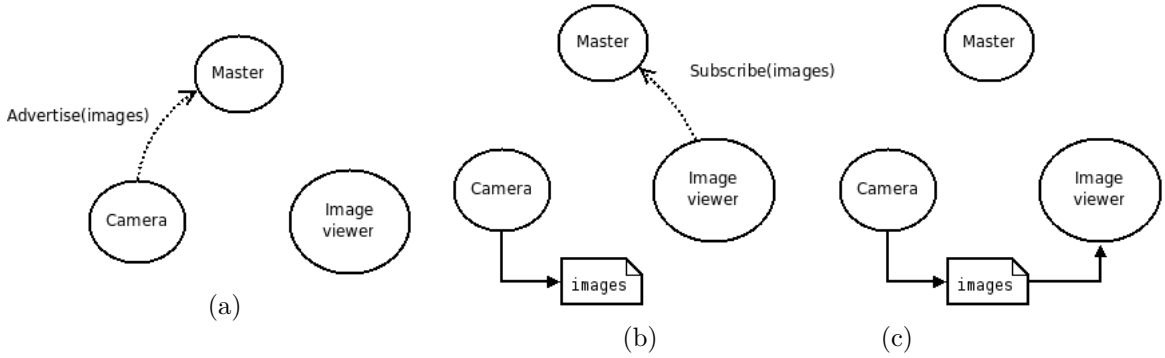


Figure 7: Conventional encryption and Fully Homomorphic Encryption (FHE)

5.3 RViz

The simulation of the encrypted formation control of the Nexus robot is performed in the 3D-visualization program RViz (ROS Visualization). The robots that are used in the formation can be loaded in this environment and the movements and behavior of theses robots can be monitored real-time. In Figure 8 three robots in formation can be seen simulated in RViz [24].

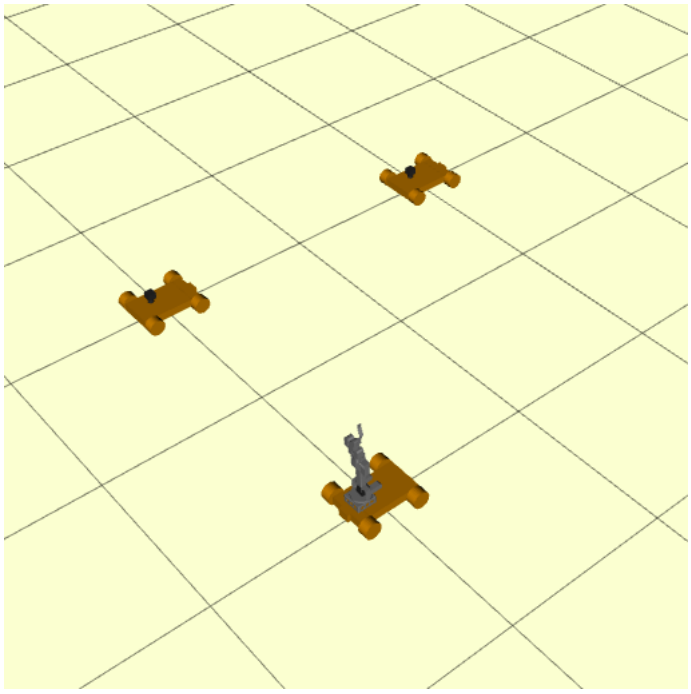


Figure 8: The Nexus formation in the 3D-visualization program RViz

5.4 Scripts

The algorithm that is developed to implement homomorphic encryption scheme in the formation control algorithm, is composed of three scripts that run separately. The first script senses the coordinates of the robots involved in the formation. These coordinates are combined in

an array and scaled with a number of the power 10. This is done to overcome the problem of working with integers as explained in chapter 3. This data is then encrypted and published to the topic 'Array'.
The second script subscribes to this topic and retrieves the encrypted data. Besides the data received from the topic 'Array', this script also receives the input data from the user, which is the encrypted velocity for the leader robot. This second scripts is the controller which performs the arithmetic operations on the encrypted data. After the arithmetic operations are performed the, still encrypted, data is published to the topic 'Inputs'.
The third and final scripts is subscribed to the topic 'Inputs' and therefore receives the encrypted data. The encrypted data is then decrypted and transformed into a format that can be sent to the actuators of the Nexus robots. Eventually a velocity is sent to the robots that is related to the desired shape of the robot formation.
The program is divided into these three scripts such that the first and last script can be programmed at the Nexus robots and the second script, the controller, can serve as the cloud. In this way the structure which can be seen in Figure 9 can be obtained.

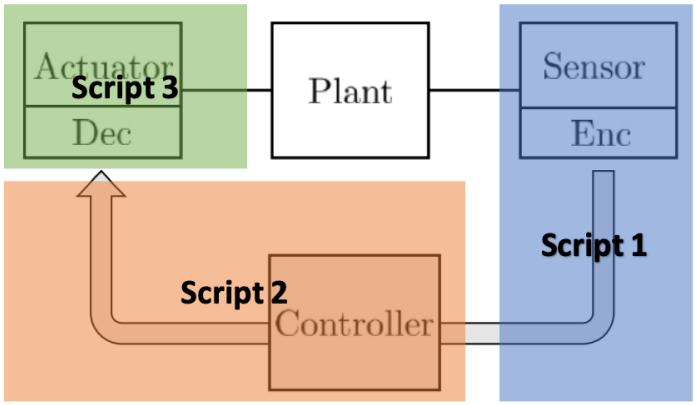


Figure 9: Functions executed by scripts

Custom message
As explained earlier, ROS supports different messages types to be used for the communication over topics. Besides this, some messages contain a special message type called a 'Header'. The header contains some common metadata fields such as a timestamp and a frame ID. These metadata is important for the formation control, since this timestamp tells when certain data, such as a position of a robot or a scan with a laser scanner, is acquired. When the data of the position of the robot is encrypted and stored in an array, the timestamp is lost since this is not present in a standard array message. In the controller, two encrypted data arrays are received, one from the formation with encrypted positions and one from the user with encrypted data inputs for the robot leader. To make the controller process the correct data that is related to the same timestamp a custom message is created. This custom message is a standard Int64MultiArray message, however a timestamp is implemented in this custom message which makes it possible for the controller to process the data appropriately.

6 Simulation

This chapter includes some simulations that were performed. First the effect of different scaling factors is simulated and discussed. After this, the trajectory of different shaped formations are analyzed and the related inter-agent distances are discussed. Based on the parameters that are used during the simulations the security level of the simulations is discussed. Finally the chapter contains the results of simulations that are performed multiple times. After this the root mean square error is compared and the robustness of the system is discussed.

6.1 Scaling factor

As explained in chapter 3 the scaling factor can be used when digits numbers are important for the level of accuracy that is required. The simulation of the formation control algorithm uses the coordinate system to determine the relative positions to the leader robot. The amount of digits that is required of these coordinates is analyzed based on simulations with different scaling factors. The results of this analysis can be seen in Figure 10, where the inter-distances for all the robots are plotted during the simulation with scaling factors 1, 10, 100 and 1000.

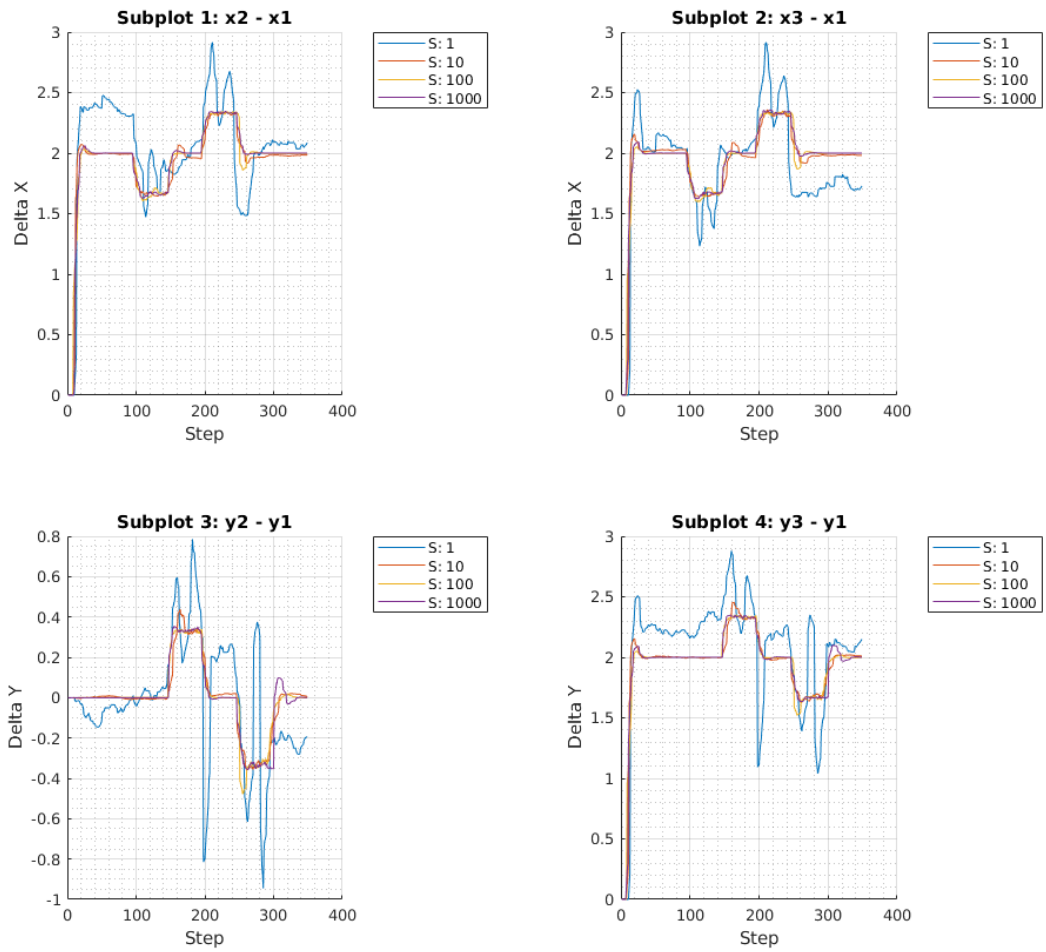


Figure 10: Inter-agent distances based on different scaling factors

It can be seen that a scaling factor of 1 does not have a sufficient accuracy, since the inter-distances between the different robots is fluctuating a lot compared to the other scaling factors. The rounding operation in the decryption of the encrypted coordinates has a large impact and makes the message after decryption differ a lot from the original coordinate message. The related trajectory can be seen in Figure 11, which also shows the large deviations in inter-distances. Even though the inter-distances do no vary a lot when looking at the scaling factors 10, 100 and 1000 the scaling factor of 1000 will be used for the rest of the simulations since this results in higher level of accuracy for the decrypted coordinate messages.

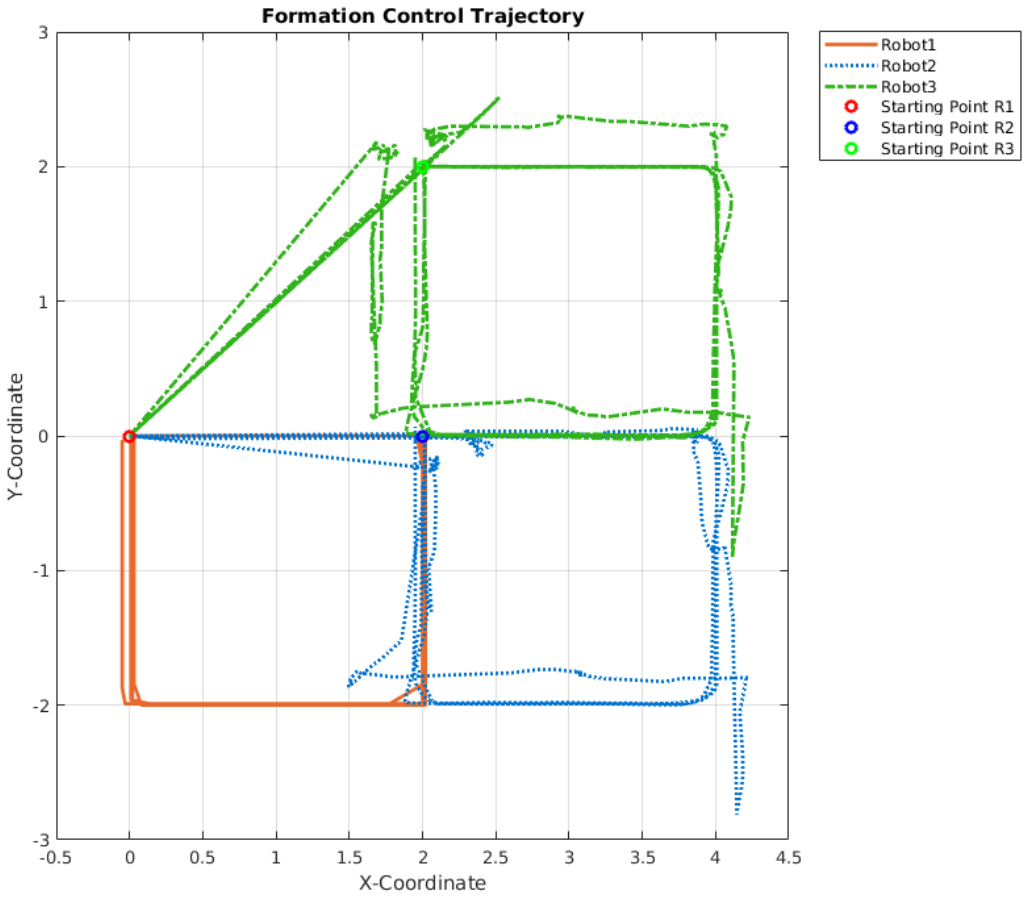


Figure 11: Trajectory of the formation based on different scaling factors

6.2 Trajectory

In Figure 12 the trajectory of a formation in the shape of a triangle can be seen. This formation trajectory is simulated in RViz and all the robots load in the (0, 0) coordinate point. When the formation control algorithm is started the two following robots are forced to their desired position based on the distance to the leader robot. In this case the desired position of the following robots are (2,0) for robot 2 and (2,2) for robot 3. These desired positions are indicated by the "Starting Point R2" for robot 2 and "Starting Point R3" for robot 3. Next, the leading robot drives along a desired route and the other two robots follow this route while maintaining their distance to the leader.

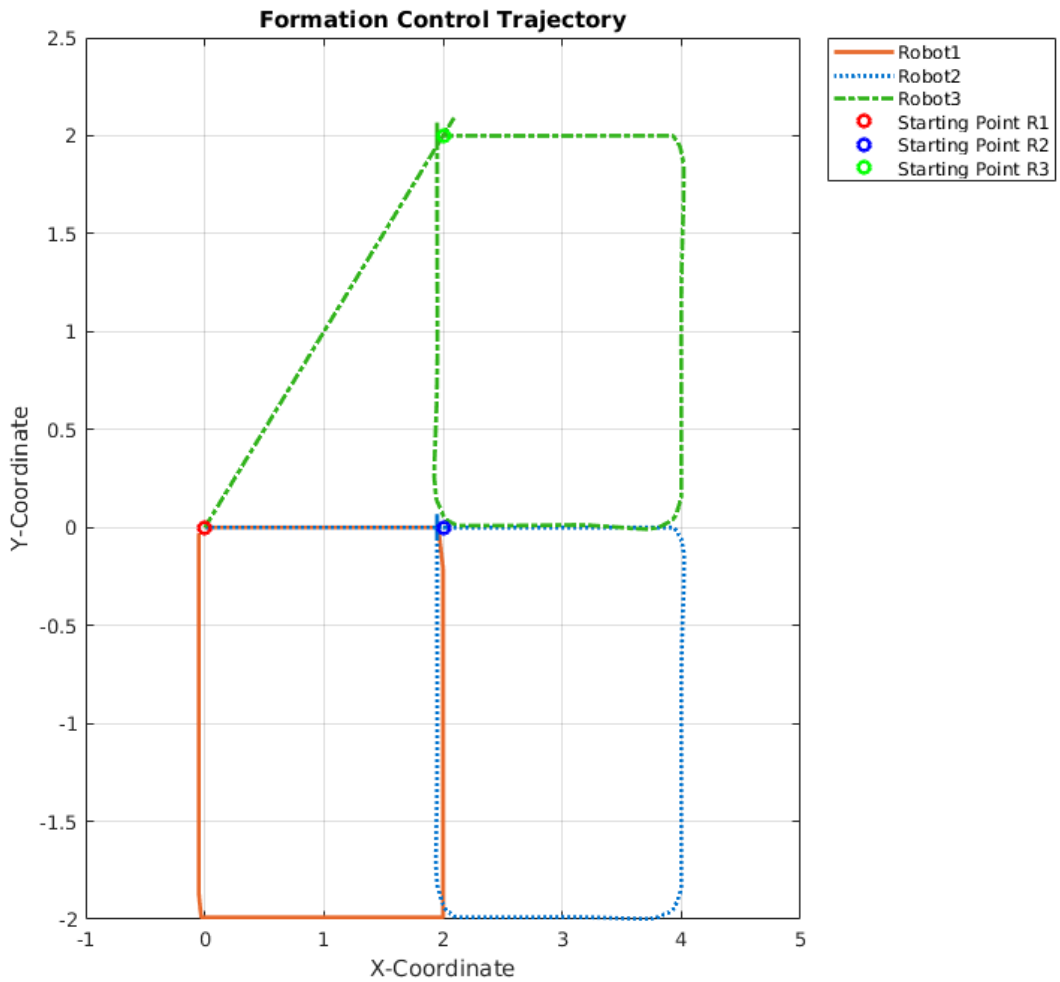


Figure 12: Trajectory of the formation in a triangle

In Figure 13 the trajectory of the second simulation can be seen. In this simulation the robots drive in a platoon, where the robot in the middle is the leading robot. The shape of the formation does not affect the behavior of the formation since it is very similar to the behavior of the triangular formation.

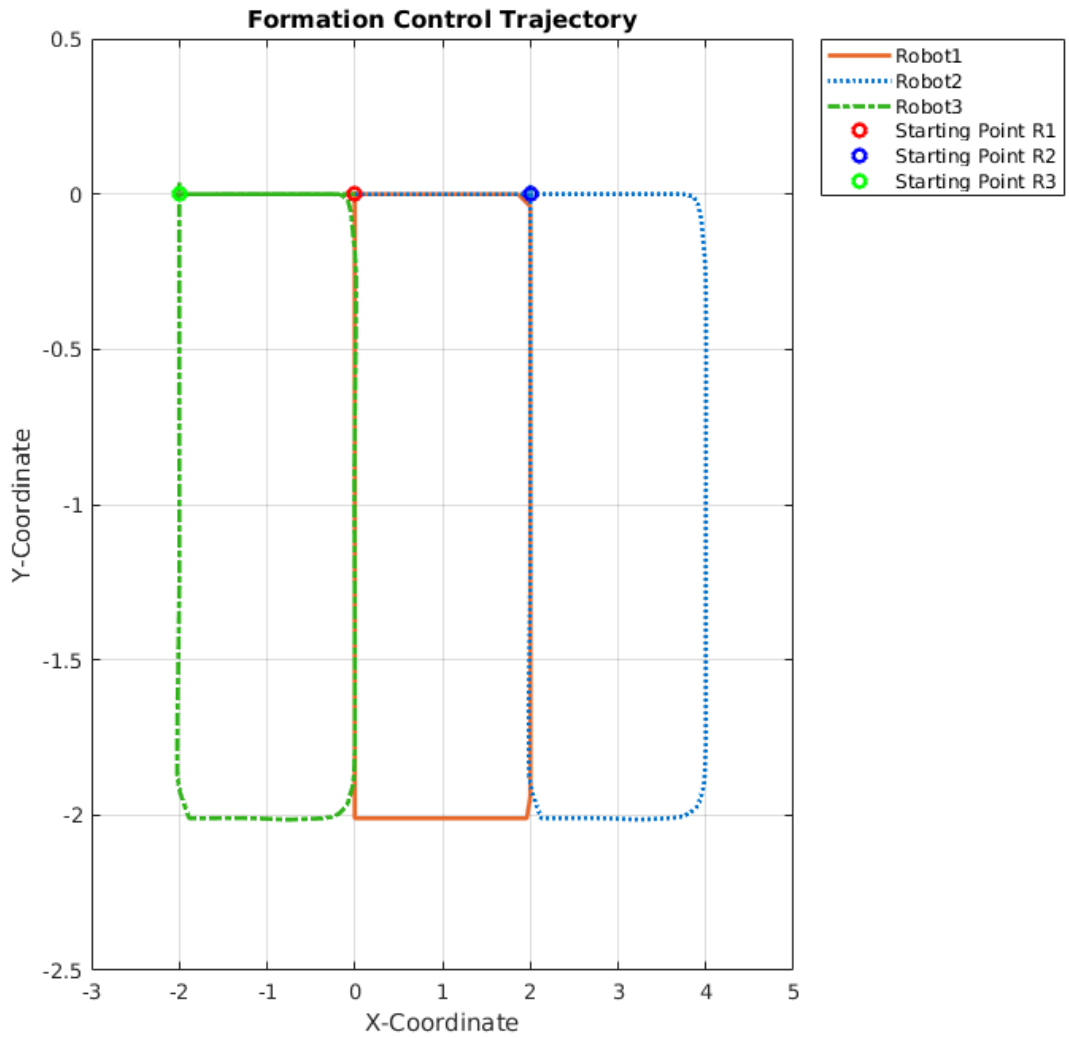


Figure 13: Trajectory of the formation in a platoon.

The previous trajectories in Figure 12 and Figure 13 show that the formation stays in the desired shape with the implementation of the homomorphic encryption scheme regardless of the type of shape. To see if the implementation of the homomorphic encryption scheme affects the shape of the formation, the same formation in a platoon is plotted in Figure 14 without the implementation of the homomorphic encryption scheme. When looking at Figure 13 and Figure 14 the trajectories are very similar. This would imply that the implementation of the homomorphic encryption scheme is successful and the behavior is identical.

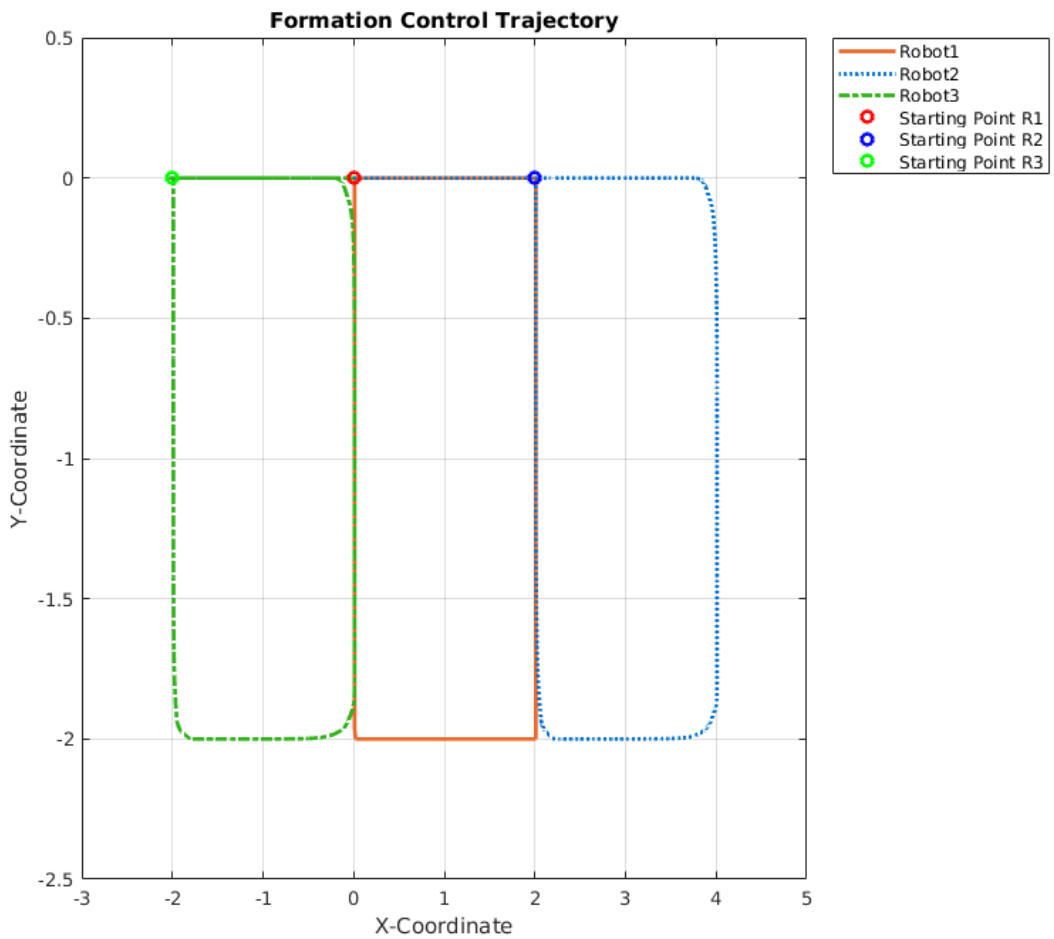


Figure 14: Trajectory of the formation in a platoon without the implementation of the homomorphic encryption scheme.

6.3 Inter-agent distances

In Figure 15 the different inter-agent distances are monitored for the simulation that is related to the trajectory in Figure 12. The different distances for the x and the y direction for robot 2 and robot 3 to the leading robot are plotted. Depending on the movement in a certain direction the distances deviate. When the the leading robot starts moving along the x axis towards the robot 2 it can be seen that the distance becomes smaller. Similarly, the distance becomes larger if the robot is moving away from the robot. The same behavior can be seen for the movement in the y direction. This is caused by a delay, caused by the computation time of the algorithm.

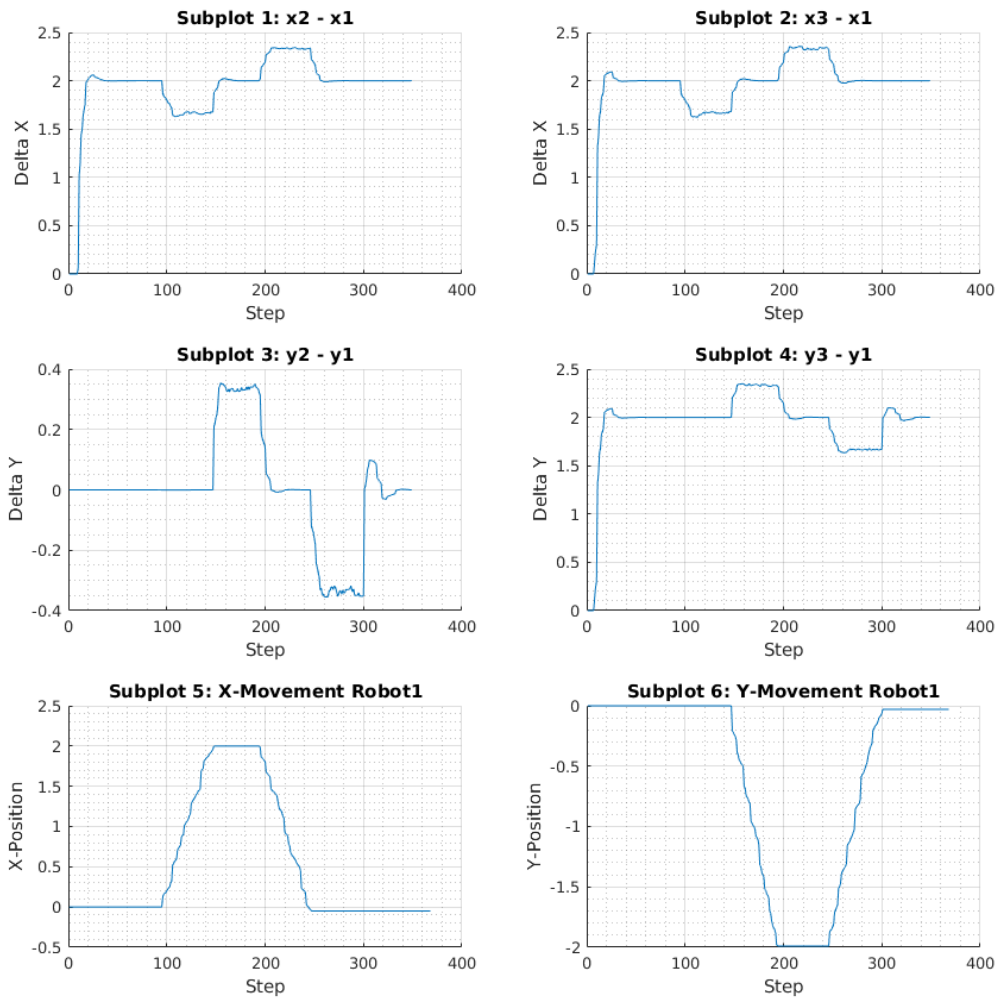


Figure 15: Distances to the leader, robot 1, and the x and y movement of robot 1 in a triangular formation.

In Figure 16 the different inter-agent distances can be seen, related to the trajectory of simulation 2 in Figure 13. In this simulation the formation was driving in a platoon instead of in a triangular formation shape. Even though the formations have different shapes, the behavior and the inter-agents distances are very similar to the ones of the formation in a triangular shape, while considering the shape of the formation.

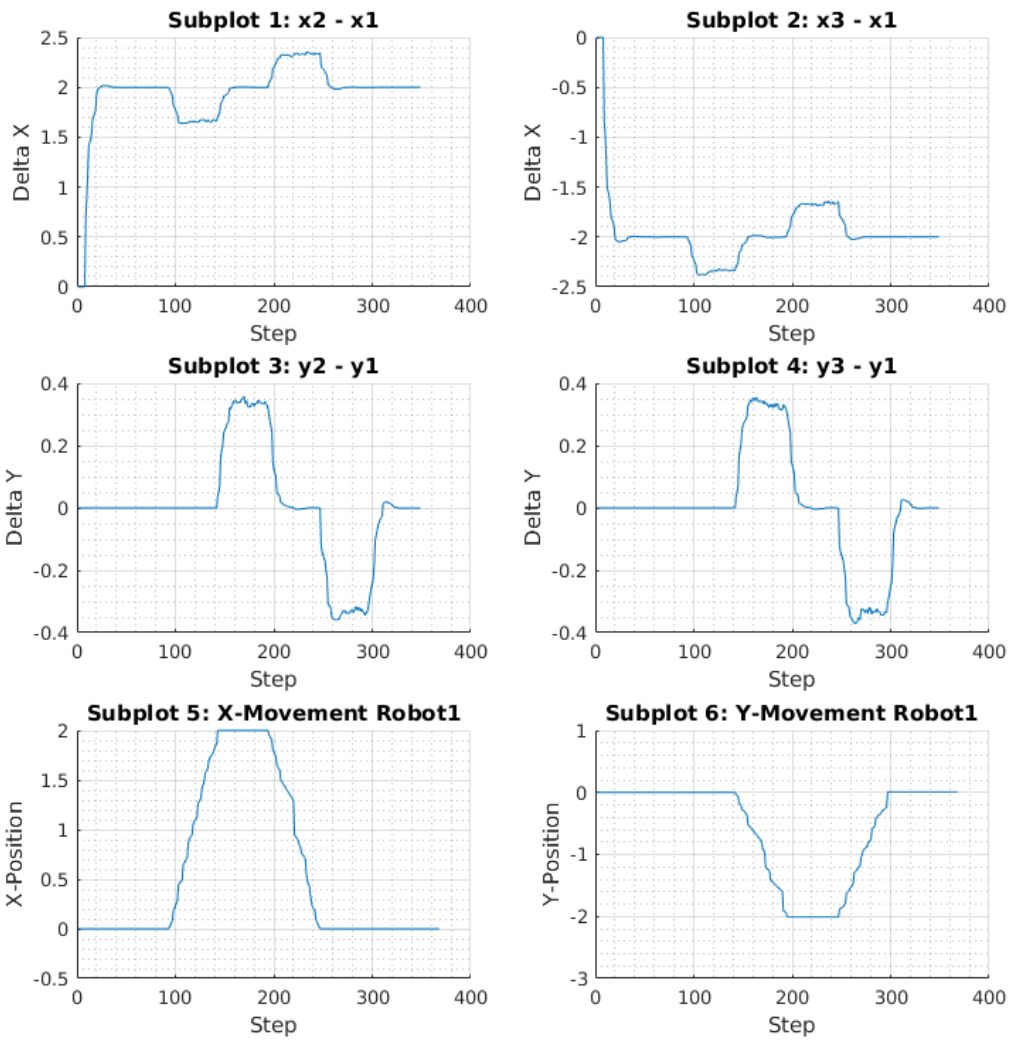


Figure 16: Distances to the leader, robot 1, and the x and y movement of robot 1 in a platoon.

The shape of the formation does not really affect the behavior of the inter-agent distances, based on Figure 16 and Figure 15. The effect of the implementation of the homomorphic encryption scheme on the inter-agent distances is also analyzed. In Figure 17 the inter-agent distances can be seen with the same platoon formation as in Figure 13, however in this simulation the encryption scheme was not used. The plots show that the behavior is almost identical, with the same movement the inter-agent distances behave the same. This shows that the implementation of the homomorphic encryption scheme does not affect the behavior of the formation control algorithm. The small errors in the inter-agent distances can be explained as normal behavior of a formation control algorithm and are caused by small delays due to computation times.

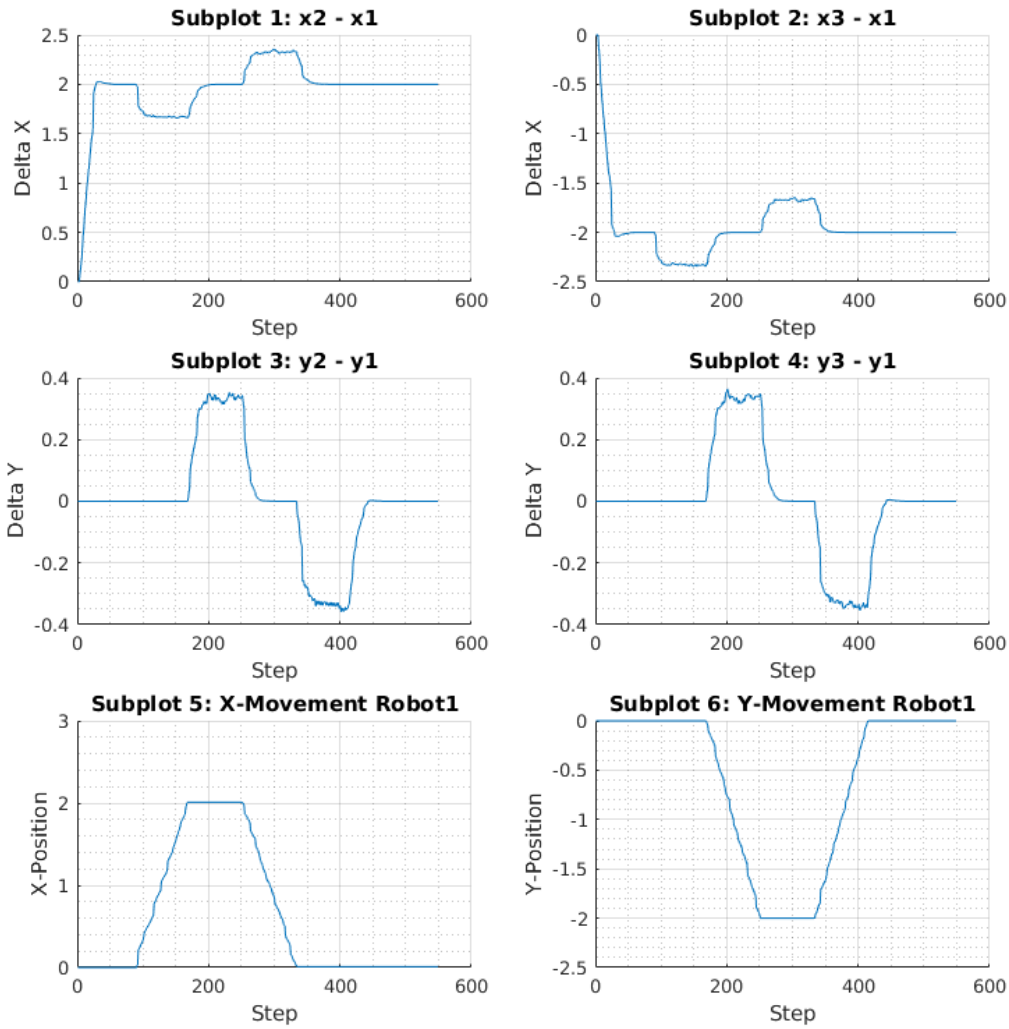


Figure 17: Distances to the leader, robot 1, and the x and y movement of robot 1 without encryption in a platoon.

6.4 Multiple Simulations

In the previous sections of this chapter, different plots of simulations have been discussed. The result suggests that the implementation of the homomorphic encryption scheme does not affect the behavior of the formation. These plots only contain the results of a single simulation and to be able to draw conclusions, a single simulation is not enough. This section is therefore dedicated to more extended simulations in which multiple scenario's are simulated many times. After these simulations the results can be statistically analyzed and better conclusions can then be drawn from the obtained results. The results of the simulations are used to analyze the data based on a 95 percent confidence interval. This 95 percent confidence interval displays an area where there is a 95 percent probability that the results of a simulation will be within this confidence bound.

In Table 2 the different scenario's that will be used during the simulations can be seen. During the simulation, the leader robot will follow a trajectory and the other two robots will follow while maintaining the desired shape of the formation. During the simulations the desired inter-agent distances will vary, for two meter inter-agent distance for scenario 1 and four meter inter-agent distance for scenario 2. Both these scenario's will be simulated with different velocity's, 0.2, 0.4 and 0.6 m/s. This results in six scenario's that will be simulated and every simulation will be performed 100 times. Besides this, the simulations will be performed two times, with the homomorphic encryption scheme implemented and without the homomorphic encryption scheme implemented. This results in a total of 1200 simulations, which must give enough results to analyze the effect of the implementation of the homomorphic encryption scheme in the formation control algorithm.

Description	Scenario 1	Scenario 2
Inter-agent Distance	2 m	4 m
Velocity of leader robot	0.2, 0.4, 0.6 m/s	0.2, 0.4, 0.6 m/s

Table 2: Scenario's used for the different Monte Carlo simulations

In Figure 18 the results can be seen from the simulations of scenario 1, where the inter-agent distance is set to be 2 meter. The inter-agent distance is plotted for leader robot 1 and robot 2. The plots in the left column show the results of the inter-agent distance with encryption implemented for the three different velocity's. The results of the same trajectory with the same velocity's are plotted in the right column, however during these simulations the data was not encrypted.
Based on the plots, it can be seen that the behavior for the situation under encryption and without encryption is very similar. The inter-agent distances behave the same and have similar maximum and minimum values. Besides this, it can be seen that the velocity does influence the inter-agent distances. During the simulations in which the leader robot has a larger velocity, the minimum and maximum values of the inter-agent distances are much higher.
Besides this, the simulations in which the data is encrypted show a small overshot when the formation is settling. This can be explained by the different frequency's at which the different scripts publish there data. In the situation of the encrypted data the frequency at the first script that senses the coordinates of the robots, is much lower compared to the situation

without encrypted data. If the frequency's would be the same for the different scripts, the data that is sent from the controller to the robot will pile up. The frequency at which the robot sends the sensed data to the controller must therefore be much lower. This results in less data points under encryption which results in a overshoot in the plots. This also explains the variation in the length of the x-axis of the plots. Without encryption much more data can be processed and therefore stored.

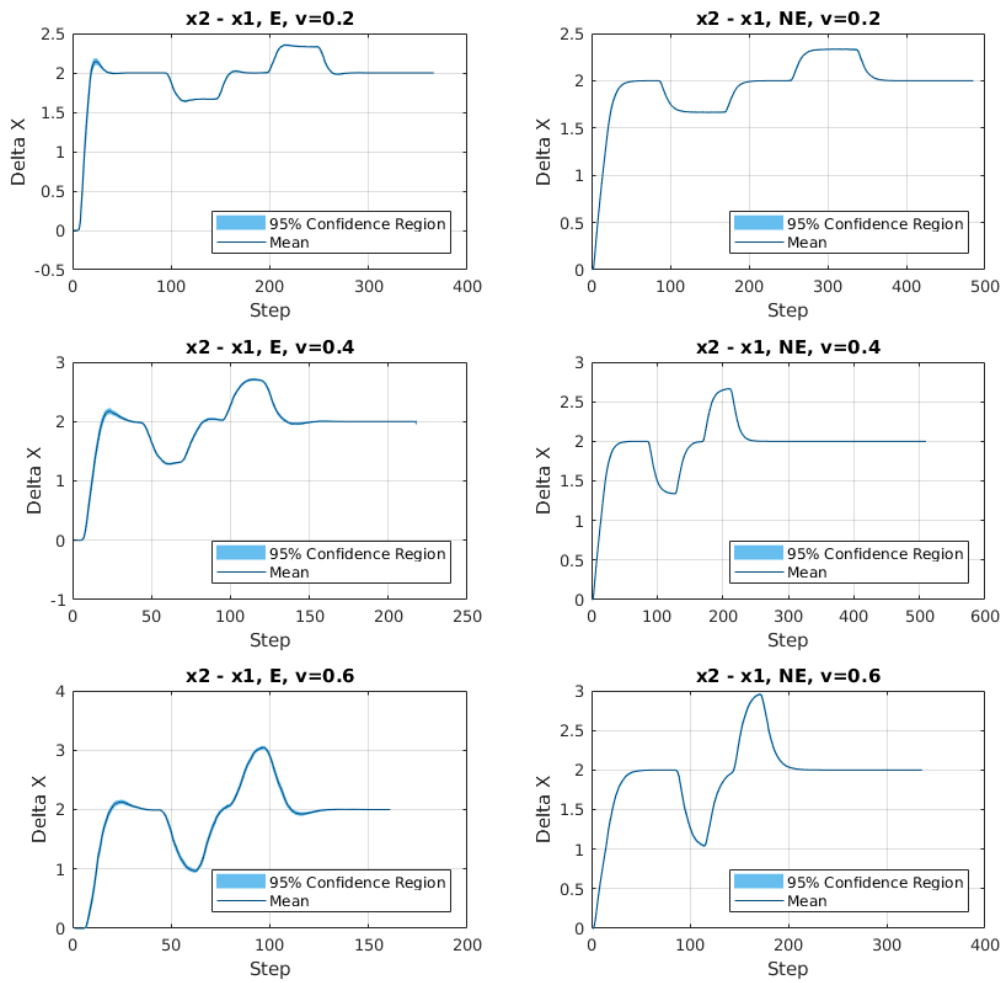


Figure 18: Inter-distance between robot 2 and leader robot 1 on x-axis for scenario 1 with encryption "E" and without encryption "NE"

Similar for scenario two, where the inter-agent distance is set to be 4 meter, the inter-agent distances between robot 2 and robot 1 are plotted. Even though the desired inter-agent distance is increased, the maximum and minimum value of the inter-agent distance are very similar compared to the plots of the desired inter-agent distance of 2 meter. The same holds for the overshoot, which again can be explained by the difference in frequencies used in the different scripts. Based on the results in Figure 18 and 19 it can be concluded that the implementation of the homomorphic encryption scheme in the formation control algorithm is successful and does not really affect the behavior of the formation.

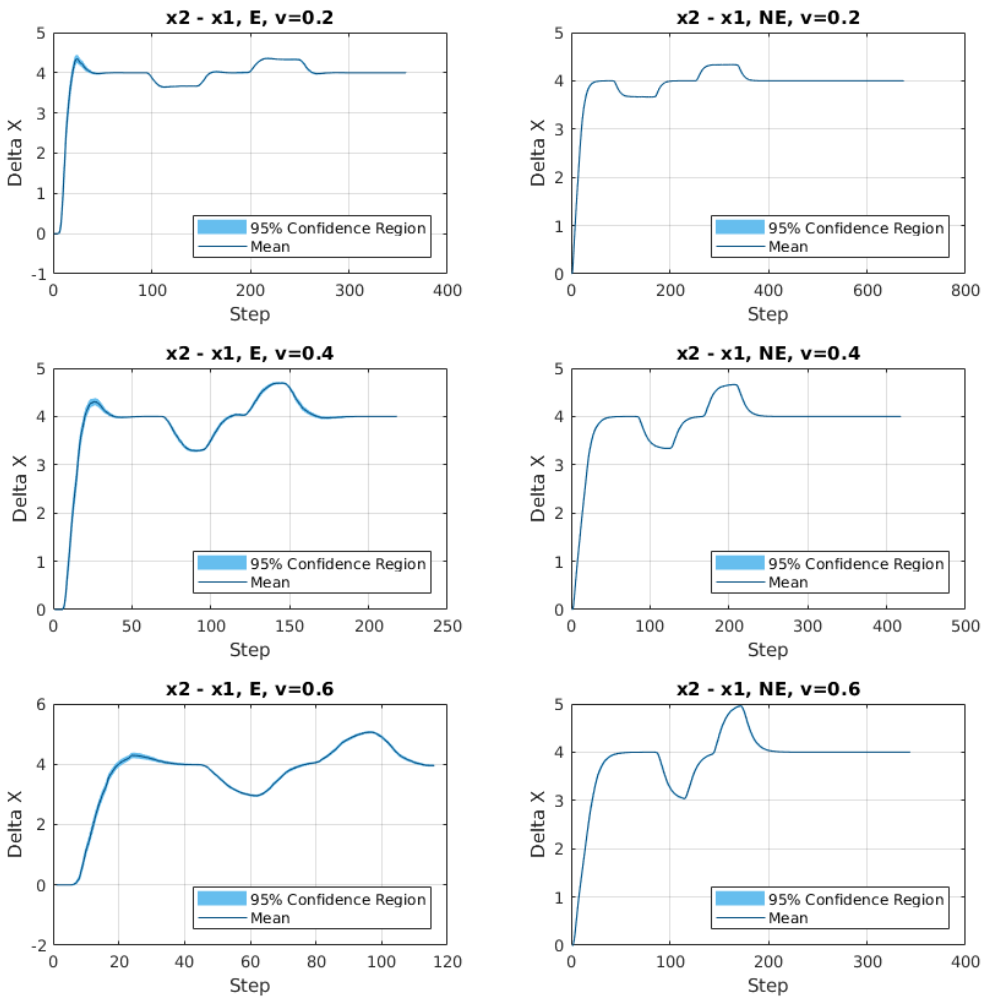


Figure 19: Inter-distance between robot 2 and leader robot 1 on x-axis for scenario 2 with encryption "E" and without encryption "NE"

6.5 Robustness of the system

To further quantify the robustness of the system and the affect of the implemented homomorphic encryption scheme, the root mean square error is computed for the different scenario's with and without encryption.

RMSE: Root Mean Square Error
The root mean square error can be used to determine how much the formation deviates from the desired shape. The inter-agent distances in the two scenario's are analyzed based on the root mean square error. The inter-agent distance between one of the following robots and the leader robot is measured on either the x- or the y-axis. The distances is then compared with the desired distance, after which the error can be determined. The root mean square errors are then compared for the two scenario's with encrypted data and without encrypted data. The root mean square error function is defined as:

$$\text{RMSE} = \frac{1}{m}\sum_{j=1}^{m}\sqrt{\frac{1}{n}\sum_{k=1}^{n}(y_k - \hat{y}_k)^2} \tag{24}$$

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.1462	0.2952	0.4530	0.5132	0.8891	1.7240

Table 3: Scenario's used in the simulations compared to root mean square error with encryption for the distance between robot 1 and 2 on the x-axis

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.1160	0.1365	0.2410	0.2597	0.4509	0.5793

Table 4: Scenario's used in the simulations compared to root mean square error without encryption for the distance between robot 1 and 2 on the x-axis

Based on the results in Table 3 and Table 4 in can be concluded that the RMSE increases when the velocity increases. This is due to a delay caused by the computations of the input velocities based on the changing inter-agent distances. The impact of this delay is larger for higher velocities.
The larger RMSE between scenario 1 and 2 is caused by the first steps of the simulation, where the robots have to move from their initial position (0,0) to the desired position based on the desired shape. The impact of the root mean square error computed as the $y_k - \hat{y}_k$ has a larger error for $y_k = 0$ where $\hat{y}_k = 4$ instead of $\hat{y}_k = 2$. Besides this, the RMSE for the simulations under encryption are very similar to the RMSE for the simulations without encryption. These results clearly indicate the succesfull implementation of the homomorphic encryption scheme in the formation control algorithm.

Similar results were obtained for the RMSE when the y-axis is considered. The plot of the inter-agent distance between robot 1 and robot 2 can be seen in the Appendix in Figure 20 and Figure 21. The RMSE related to the different scenario's can be seen in Table 5 for the formation with encrypted data and in Table 6 for the formation without encrypted data. The desired distance between robot 1 and robot 2 on the y-axis is set to zero and the inter-agent distance on the x-axis is set to two meters for scenario 1 and to four meter for scenario 2.

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.0287	0.0930	0.1841	0.0288	0.0915	0.1727

Table 5: Scenario's used in the simulations compared to root mean square error with encryption for the distance between robot 1 and 2 on the y-axis

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.0324	0.0579	0.1154	0.0244	0.0691	0.1155

Table 6: Scenario's used in the simulations compared to root mean square error without encryption for the distance between robot 1 and 2 on the y-axis

The RMSE for scenario 1 and scenario 2 are very similar, which is expected since the desired inter-agent distance is set to zero for the y-axis in both scenario's. The RMSE increases when the velocity becomes larger, which is caused by the delay due to computation times. The differences in RMSE for the encrypted situation compared to the situation without encrypted data are again very small and can again be explained by the different frequencies used.

In Table 7 the RMSE for the formation with encrypted data can be seen for the inter-agent distance between robot 1 and robot 3 on the y-axis. The same data for the formation without encrypted data can be seen in Table 8 and the inter-agent distances are plotted in Figure 22 and Figure 23 in the Appendix. The result are very similar to the results obtained for the inter-agent distance between robot 1 and robot 2 on the x-axis. This shows that the implementation of the homomorphic encryption scheme is successful for movement along the x- and the y-axis for both the following robots.

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.1450	0.2960	0.4642	0.5232	0.8774	1.2451

Table 7: Scenario's used in the simulations compared to root mean square error with encryption for the distance between robot 1 and 3 on the y-axis

Scenario	1			2		
Inter-agent distance (m)		2			4	
Velocity (m/s)	0.2	0.4	0.6	0.2	0.4	0.6
RMSE (m)	0.1158	0.1752	0.2716	0.3173	0.5368	0.7996

Table 8: Scenario's used in the simulations compared to root mean square error without encryption for the distance between robot 1 and 3 on the y-axis

6.6 Security level of simulation

In chapter 3 the security level of the LWE-encryption scheme is discussed based on the different parameters that were chosen. In the simulations that are performed in this chapter the following values were chosen for the different parameters:

$$p = 2\hat{}45; \quad L = 256; \quad r = 64; \quad N = 999;$$

This results in the following security values after running the online LWE-estimator program:

$$\begin{aligned} \text{usvp: rop:} &= 2\hat{}86.4, \ldots \\ \text{dec: rop:} &= 2\hat{}90.5, \ldots \\ \text{dual: rop:} &= 2\hat{}91.4, \ldots \end{aligned}$$

This shows that the LWE-based encryption scheme that is used during the simulations has at least a 86.4-bit security level.

Besides this security level, the extra security feature with the introduction of the security number in the LWE-based encryption scheme is tested during the simulations. This addition to the security level of the LWE-based encryption scheme is tested with an attack algorithm. This algorithm generates an array of random integers with the dimensions of the array of the ciphertext **m** and sends this to the controller. Without the extra security feature the formation will be distorted and the follower robots will move out of formation, which could be seen during simulations. The simulations also show that the extra security feature works correctly. When the formation is under attack by the attack algorithm the security feature detects the mismatch in the security number and the formation is stopped and the warning message "Formation under attack! Stop!" is published.

7 Discussion

In the previous chapter multiple experiments were performed on the formation control algorithm with the homomorphic encryption scheme implemented. The performance of the formation control algorithm with the homomorphic encryption scheme implemented was compared to the performance of the formation control algorithm without the homomorphic encryption scheme implemented. The experiments show very similar results for the two formation control algorithms. When comparing the RMSE the formation control algorithm without encryption performance slightly better. This can be explained by the different frequencies that are used in the scripts. Since the normal formation control algorithm consists of one script, the update frequency can be set higher. The control algorithm under encryption used lower update frequencies to prevent the data from pilling up. The input variables are less frequently updated for the formation control algorithm under encryption, which results in some small overshoots.

When looking at the RMSE for the two scenario's some differences in performance can be noticed. This is caused by the coordinate point where the robots are loaded in the simulation environment in RViz. The three robots are loaded in the coordinate (0,0) and from this position they first move to their desired position before the formation starts moving. During this time the RMSE is larger when the desired distance is larger, as in scenario 2. This may result in larger RMSE for scenario 2, since the desired inter-agent distance is set to 4 meter instead of 2 meter.

It can be argued that the formation control algorithm that is used can be improved, when the robustness of the system is analyzed. The two following robots are now only controlled by the behavior of the leader robot. This makes the formation less robust. The formation would be more robust if all the robots control each other. When one of the robots starts drifting, the other robots can influence this behavior and prevent the entire formation from drifting.

However, the introduction of a more advanced formation control algorithm will have to overcome the limitations of the homomorphic encryption scheme. The formation control that is currently used consists of simple arithmetic operations, more advanced formation control algorithms often require more complex operations. Complex operations, such as an Euclidian which is commonly used in formation control, are currently not supported in the homomorphic encryption scheme.

The formation control algorithm under encryption is only tested with a simulated robot formation in RViz. The implementation on the actual Nexus robots requires a different approach for sensing the positions of the robots. The positions of the robot are now sensed in the global coordinate system, however this is not representative for a real world simulation.

8 Conclusion

The goal for this research was to implement a homomorphic encyrption scheme in the Nexus formation control to increase security and privacy. It can be concluded that this goal is met, since a homomorphic encryption scheme is successfully implemented in the formation control algorithm. The results obtained from different simulations show that the behavior of the formation is very similar for the formation control algorithm with and without the homomorphic encryption scheme implemented. The small differences in the RMSE can be explained by the starting positions of the robots in the coordinate (0,0), which makes the RMSE larger for a desired distance of 4 meters compared to 2 meters. The small overshoots can be explained by the different frequencies used in the scripts. Apart from these small differences, the results and performance of the two formation control algorithm are very similar. Besides this the formation control algorithm has become more secure and private by addition of the user-identification algorithm. The combination of the homomorphic encryption scheme and the user-identification algorithm has substantially increased the security and the privacy of the Nexus formation. However the formation control algorithm under encryption is only tested on a simulated formation in RViz. The implementation on an actual Nexus robot requires other techniques to sense the positions of the robots, e.g. with laser scanners.

9 Further research

Based on the results of this research, some studies for further research can be proposed. First, a study can be conducted on the implementation of distance measurement sensors that can be used in real world application. The formation is now simulated in the simulation program RViz, in which the coordinates of the robots are always known by the system. In a real world application the inter-agent distances cannot be determined based on coordinates. In this case the formation control algorithm must determine the inter-agent distances based on measurement of a laser sensor or based on angles and object detection. However, with these techniques other problems may arise such as noise in the data that can affect the performance.

Other improvements to the system could be the implementation of a rotation element in the formation control. The formation control algorithm is now designed in such a way that the robots can only move linearly along the x and y-axes. Implementing a rotation to the formation control algorithm could increase the possibilities of this system.

Besides this, the formation control law that is currently used lacks some robustness due to the existence of a explicit leader. The entire formation will be affected if the leader starts drifting, for example. Other formation control laws that do not have a explicit leader are more robust since they control each other. The more advanced mathematical operations that are introduced with more advanced formation control algorithms, may face the limitations of the homomorphic encryption scheme.

Finally, the effect of the changed frequencies can be further research before the scripts are implemented in the real Nexus robots. The update frequencies that are now used might not be feasible with the hardware that is used in the Nexus robots. When these frequencies are adjusted for the actual Nexus robot the performance of the formation control algorithm under encryption might be affected.

References

[1] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175–180, 2016.

[2] J. H. Cheon, K. Han, S.-M. Hong, H. J. Kim, J. Kim, S. Kim, H. Seo, H. Shim, and Y. Song, "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE Access*, vol. 6, pp. 24 325–24 339, 2018.

[3] S. Jafarnejad, L. Codeca, W. Bronzi, R. Frank, and T. Engel, "A car hacking experiment: When connectivity meets vulnerability," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6.

[4] H. Sandberg, S. Amin, and K. H. Johansson, "Cyberphysical security in networked control systems: An introduction to the issue," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 20–23, 2015.

[5] F. Farokhi, I. Shames, and N. Batterham, "Secure and private cloud-based control using semi-homomorphic encryption," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 163–168, 2016.

[6] A. Bouti and J. Keller, "Towards practical homomorphic encryption in cloud computing," in *2015 IEEE Fourth Symposium on Network Cloud Computing and Applications (NCCA)*. IEEE, 2015, pp. 67–74.

[7] A. Loria, J. Dasdemir, and N. A. Jarquin, "Leader–follower formation and tracking control of mobile robots along straight paths," *IEEE transactions on control systems technology*, vol. 24, no. 2, pp. 727–732, 2015.

[8] Y. Li, C. Tang, K. Li, X. He, S. Peeta, and Y. Wang, "Consensus-based cooperative control for multi-platoon under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 2220–2229, 2018.

[9] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[12] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *Stoc*, vol. 9, no. 2009, 2009, pp. 169–178.

[13] A. B. Alexandru, K. Gatsis, Y. Shoukry, S. A. Seshia, P. Tabuada, and G. J. Pappas, "Cloud-based quadratic optimization with partially homomorphic encryption," *arXiv preprint arXiv:1809.02267*, 2018.

[14] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 79, 2018.

[15] D. Hrestak and S. Picek, "Homomorphic encryption in the cloud," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2014, pp. 1400–1404.

[16] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424–440, 2015.

[17] X. Dong, Y. Zhou, Z. Ren, and Y. Zhong, "Time-varying formation control for unmanned aerial vehicles with switching interaction topologies," *Control Engineering Practice*, vol. 46, pp. 26–36, 2016.

[18] J. Kim, H. Shim, and K. Han, "Comprehensive introduction to fully homomorphic encryption for dynamic feedback controller via lwe-based cryptosystem," *arXiv preprint arXiv:1904.08025*, 2019.

[19] P. Lu, W. Yu, and F. Zhang, "Leader-follower formation control with mismatched compasses," in *2017 36th Chinese Control Conference (CCC)*. IEEE, 2017, pp. 8821–8826.

[20] "ROS Nodes," http://wiki.ros.org/Nodes, accessed: 2019-07-31.

[21] "ROS Messages," http://wiki.ros.org/msg, accessed: 2019-07-31.

[22] "ROS Topics," http://wiki.ros.org/Topics, accessed: 2019-07-31.

[23] "ROS Master," http://wiki.ros.org/Master, accessed: 2019-07-31.

[24] "RViz," http://wiki.ros.org/rviz, accessed: 2019-07-31.

A Appendix

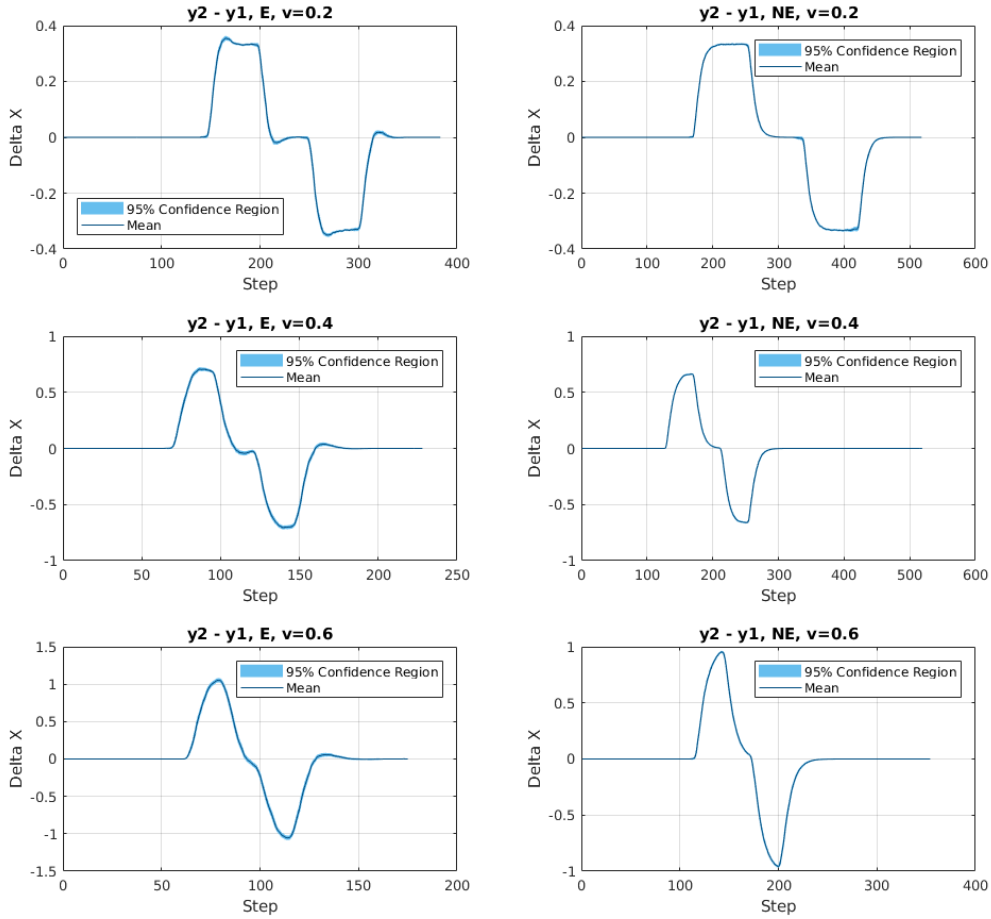


Figure 20: Inter-distance between robot 2 and leader robot 1 on y-axis for scenario 1 with encryption "E" and without encryption "NE"

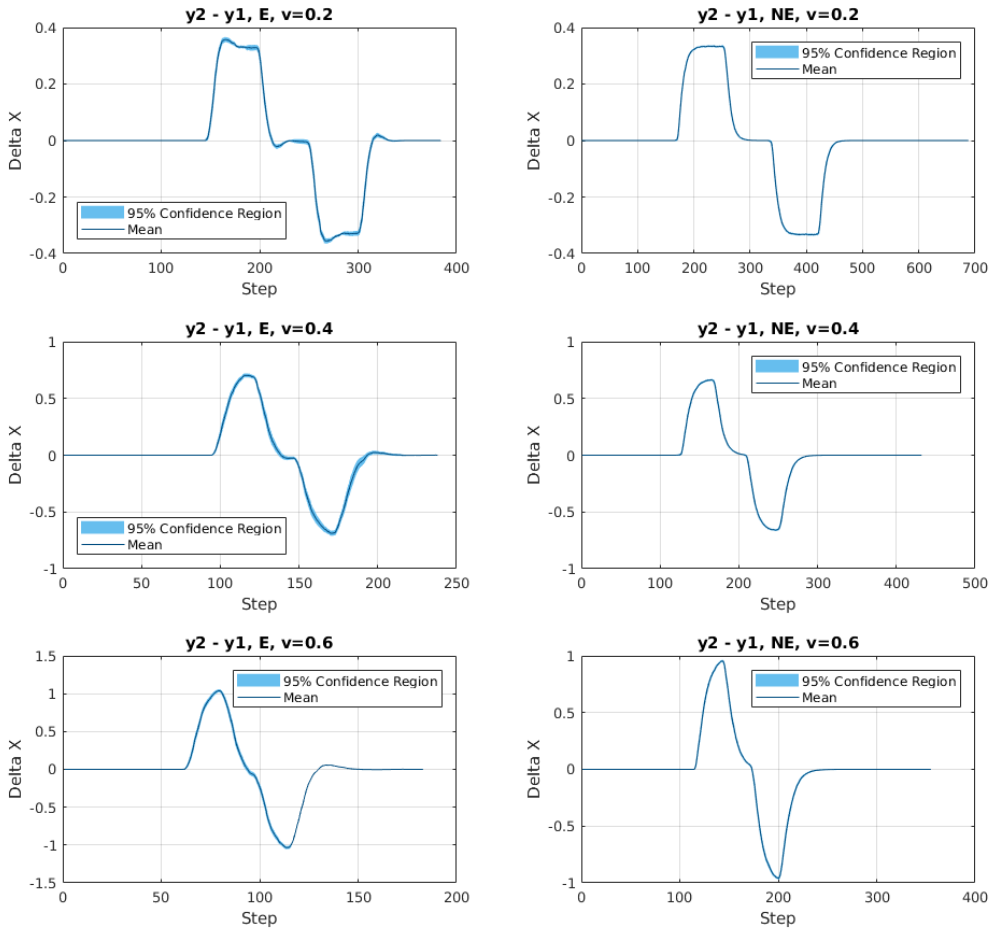


Figure 21: Inter-distance between robot 2 and leader robot 1 on y-axis for scenario 2 with encryption "E" and without encryption "NE"

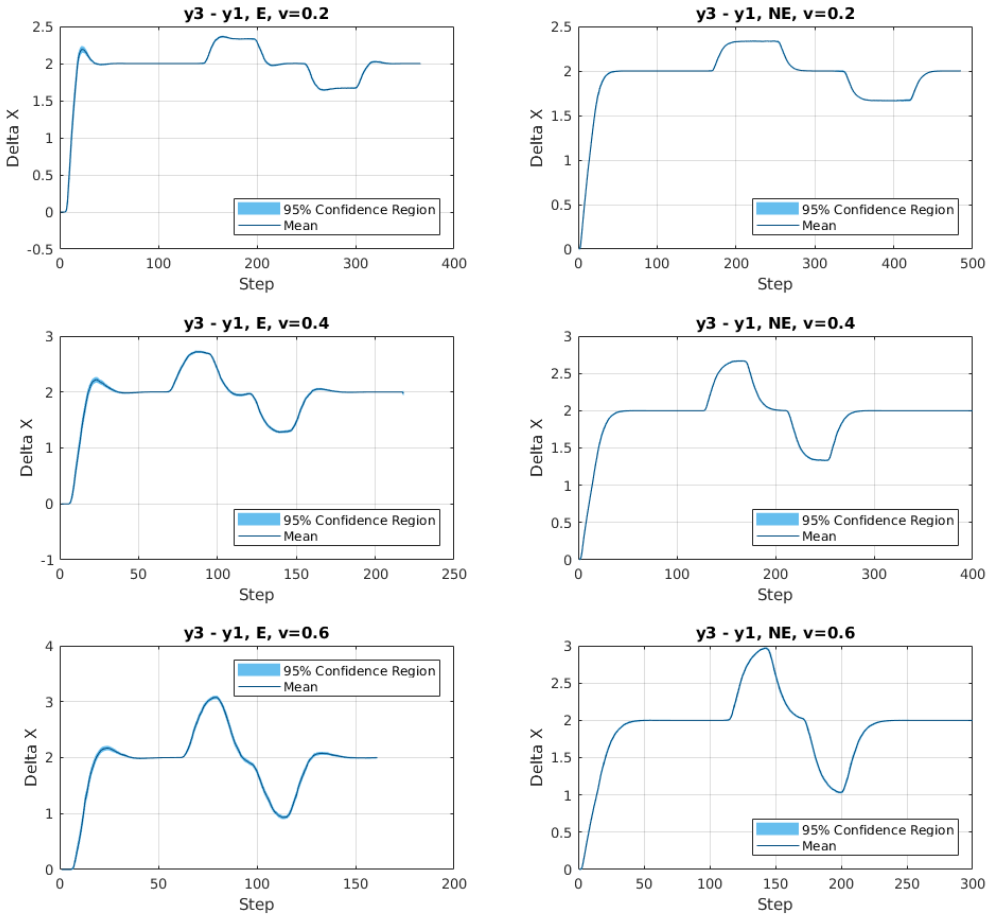


Figure 22: Inter-distance between robot 3 and leader robot 1 on y-axis for scenario 1 with encryption "E" and without encryption "NE"

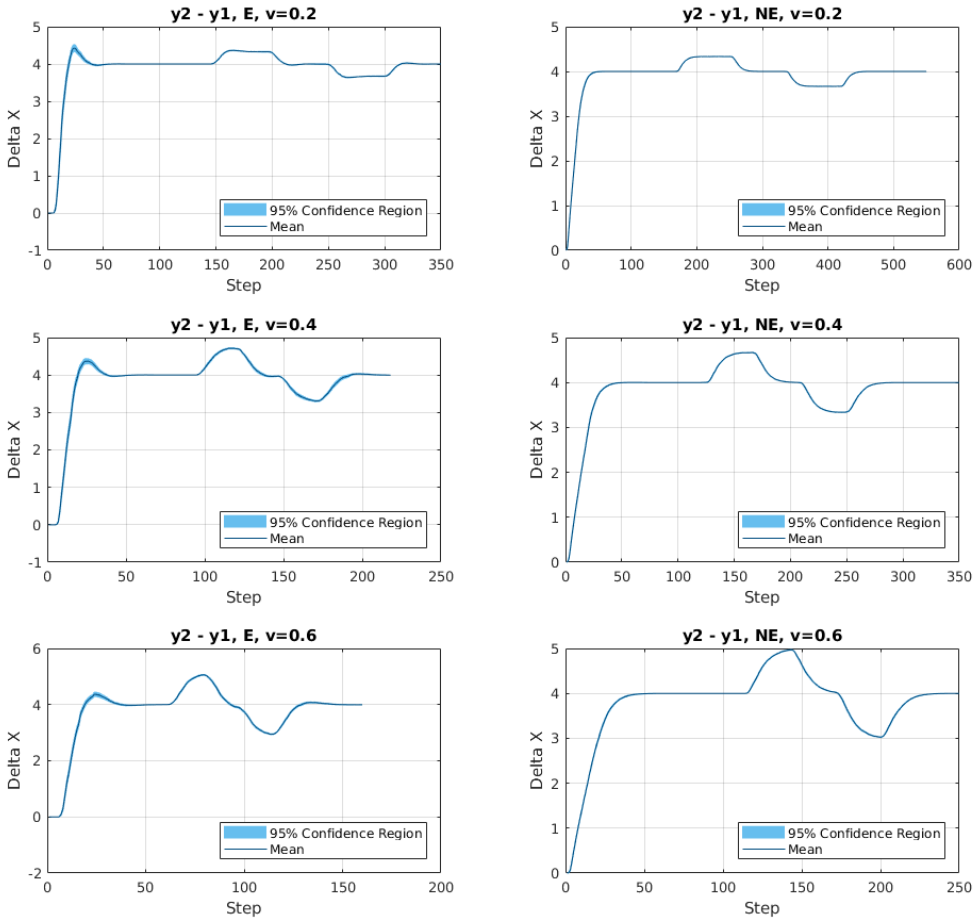


Figure 23: Inter-distance between robot 3 and leader robot 1 on y-axis for scenario 2 with encryption "E" and without encryption "NE"