FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Encrypted Swarm Coordination: Applying Paillier Encryption to the Motion Control of Starling-Inspired Agents

Sofia Viana

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: António Pedro Aguiar

July 23, 2024

Resumo

A robótica de enxame, tal como as murmurações sincronizadas dos bandos de estorninhos, oferece um enorme potencial para sistemas robóticos descentralizados. Garantir a comunicação segura dentro destes enxames ágeis e dinâmicos é crucial. Esta dissertação investiga a integração da criptografia homomórfica de Paillier no controlo de movimento de agentes inspirados em estorninhos para aumentar a sua segurança. O estudo começa com uma análise detalhada dos mecanismos de controlo existentes, revelando as dinâmicas complexas que permitem um comportamento eficaz do enxame. Com base nesta fundação, o sistema de criptografia de Paillier é integrado, facilitando cálculos seguros enquanto mantém a eficiência e coordenação do bando. Os resultados demonstram que a comunicação encriptada pode ser implementada com sucesso, preservando tanto a integridade como o desempenho do enxame. Adicionalmente, a abordagem é escalada para múltiplas famílias de agentes, demonstrando a sua versatilidade e robustez. Estes resultados avançam o campo da robótica de enxame segura e abrem novos caminhos para futuras pesquisas em estratégias de coordenação encriptada.

Abstract

Swarm robotics, much like the synchronized murmurations of starling flocks, offers immense potential for decentralized robotic systems. Ensuring secure communication within these agile and dynamic swarms in many practical applications can be crucial. This dissertation investigates the integration of Paillier homomorphic encryption into the motion control of starling-inspired agents to enhance their security. The study begins with a thorough analysis of existing control mechanisms, revealing the complex dynamics that enable effective swarm behavior. Building on this foundation, the Paillier cryptosystem is integrated, facilitating secure computations while maintaining the efficiency and coordination of the flock. The results demonstrate that encrypted communication can be successfully implemented, preserving both the integrity and performance of the swarm. Additionally, the approach is scaled to multiple families of agents, showcasing its versatility and robustness. These findings advance the field of secure swarm robotics and open new pathways for future research in encrypted coordination strategies.

Agradecimentos

Gostaria de agradecer bastante a todos aqueles que, direta ou indiretamente, contribuíram para a realização desta dissertação. Este trabalho representa não apenas um marco na minha trajetória académica, mas também o culminar de anos de apoio e incentivo de várias pessoas especiais na minha vida.

Aos meus pais, agradeço por me proporcionarem todas as oportunidades que desejei e por todo o apoio incondicional ao longo dos anos.

Aos meus irmãos, Inês Viana e Guilherme Viana, por serem os meus melhores amigos e por sempre me apoiarem nos momentos mais difíceis.

Aos meus amigos, Ricardo, Rafael, Maria e Diogo, pela partilha dos desafios da vida académica e pelos inúmeros bons momentos que passámos juntos.

Ao João Brito, pelo constante apoio nos momentos em que mais precisei e por ter a paciência de lidar com todas as minhas crises existenciais.

Ao meu orientador, Professor António Pedro Aguiar, pela orientação e apoio inestimáveis nestes últimos meses.

A todos os meus professores, desde a escola primária até ao ensino superior, por me moldarem na pessoa que sou hoje e por transmitirem os conhecimentos que possuo.

A todos, o meu mais sincero agradecimento.

Sofia Viana

"Happiness can be found even in the darkest of times,
if one only remembers to turn on the light "

J.K.Rowling

Contents

List of Figures

List of Tables

Abbreviations

HE	Homomorphic Encryption
PHE	Partially Homomorphic Encryption
SHE	Somewhat Homomorphic Encryption
FHE	Fully Homomorphic Encryption
gcd	Greatest Common Divisor
lcm	Least Common Multiple

Chapter 1

Introduction

1.1 Context

Drawing parallels to how bees and starling birds fly in thousands, exhibiting perfectly coordinated movements as if guided by a single brain, swarm intelligence strives to emulate these natural phenomena, implementing decision-making and collective learning processes. Rather than relying on a single entity, it harnesses the shared intelligence of a multi-agent system, thereby offering robust, scalable, and flexible solutions for complex tasks.

Boosted by advancements in cloud computing, enhanced hardware architecture, and innovative manufacturing methodologies [9], the field of swarm robotics is poised to become the primary catalyst for innovation in the realm of robotics. As identified by [25], its potential is recognized as one of the top ten "Robotics Grand Challenges" for the forthcoming 5 to 10 years, anticipated to have a considerable socioeconomic influence. In light of this, naturally, the scope of its applications is vast, encompassing areas such as space exploration [12], disaster management [3], or urban waste management [6].

Despite its promising future, these systems have yet to transition to real-world applications. Deploying any autonomous robot [8] must ensure operational safety, data privacy, and resilience against external malicious attacks. The severity of these issues escalates in the context of robot swarms [11], where challenges such as entity authentication, data confidentiality, and integrity are intensified due to the interaction of potentially hundreds of robots.

Research in the security of robot swarms is still in its early stages, exploring both conventional (e.g., cryptographic Merkle trees [9]) and novel (blockchain [24]) security approaches for integration into the control architecture of robot swarms. Preliminary studies address privacy preservation in a swarm [9], disruption prevention from malicious robots [23], and countermeasures against Sybil attacks (when a single harmful entity creates numerous false identities to gain excessive control, consequently compromising the network's integrity) [24].

One particularly interesting approach is integrating *Homomorphic Encryption* (HE) to ensure secure communications within agents of the same system [15]. At its core, HE allows computations to be conducted directly on encrypted data, eliminating the need for decryption and preserv-

ing the confidentiality of sensitive information throughout the control process, thus holding great promise for enhancing the security and privacy of data in swarm robotics.

The research presented in this dissertation aims to explore exactly that pioneering approach. By implementing a simple simulation, we examine a scenario where a fixed number of agents navigate to a specific position while avoiding obstacles. Importantly, this is achieved while keeping the agents' positions confidential through the use of Paillier's encryption method, a type of homomorphic encryption. By incorporating this encryption into the control law of swarm robots, we ensure that each robot can operate based on shared information without compromising the privacy of individual agents' positions. This not only strengthens the system's security but also opens up new possibilities for using swarm robotics in situations where maintaining confidentiality is crucial.

1.2 Objectives

The primary objective of this dissertation is to develop and implement a simulation of swarm agents that achieve goal-directed motion control using homomorphic encryption to ensure privacy. Specifically, the objectives are itemized as follows:

1. **Develop a Comprehensive Theoretical Framework:**
    - Review and synthesize the foundational principles of cryptography, with a focus on homomorphic encryption.
    - Detail the principles and operation of Paillier's encryption scheme and its applicability to swarm robotics.
2. **Design and Implement System Architecture:**
    - Define the interaction protocols between the agents and the environment, ensuring seamless integration of the homomorphic encryption scheme.
3. **Develop Motion Control Algorithms:**
    - Implement motion control laws that enable a swarm of agents to navigate toward a goal position while avoiding obstacles.
    - Ensure that these control laws can function both with and without the integration of homomorphic encryption.
4. **Integrate Homomorphic Encryption:**
    - Adapt the developed motion control algorithms to incorporate Paillier's homomorphic encryption, maintaining the privacy of each agent's position data.
    - Validate the encrypted control system to ensure it performs comparably to the non-encrypted system in terms of efficiency and accuracy.

5. **Extend to Multiple Robot Starling Families:**
    - Adapt the control and encryption mechanisms to accommodate different families of robots, demonstrating the versatility and scalability of the approach.
6. **Analysis and Validation:**
    - Analyze the results to assess the impact of homomorphic encryption on the overall system performance, including computational overhead and communication efficiency.

The successful completion of these objectives will demonstrate the feasibility and effectiveness of using homomorphic encryption to ensure privacy in the motion control of swarm robotics, paving the way for more secure and robust implementations in real-world applications.

1.3 Contributions

This dissertation has contributed to the field of secure swarm robotics by achieving the following:

1. Synthesized relevant cryptographic principles to establish a theoretical framework for applying homomorphic encryption in swarm robotics;
2. Conducted a comprehensive review of the foundational principles of homomorphic encryption, with a specific focus on Paillier's encryption scheme;
3. Developed a system architecture that integrates homomorphic encryption into the communication protocols of swarm agents in order to ensure secure and private data exchange within the swarm;
4. Adapted motion control algorithms that enable swarm agents to navigate toward target positions while avoiding obstacles, ensuring the algorithms' functionality in both encrypted and non-encrypted scenarios;
5. Validated the performance of the encrypted system;
6. Extended the control and encryption mechanisms to accommodate different families of robots;
7. Analyzed the impact of homomorphic encryption on system performance, including computational overhead and communication efficiency;
8. Provided an assessment of the trade-offs between privacy and system performance in swarm robotic systems;
9. Identified potential research directions for enhancing security in swarm robotics.

1.3.1 Sustainable Development Goals (SGD)

The research presented in this dissertation contributes to the advancement of secure swarm robotics, which has the potential to significantly impact various global challenges. In alignment with the United Nations Sustainable Development Goals (SDGs)[1], this work addresses key areas such as hunger and inequality. The following table summarizes the specific goals, contributions, and performance indicators related to this dissertation.

SDG	Goal	Contribution	Performance indicators and metrics
(2) No Hunger	1	Swarm robotics can optimize planting processes, ensuring efficient seed distribution and pest control in agricultural fields.	- Reduction in the time taken for planting processes. - Increase in crop yields per hectare. - Decrease in the use of pesticides through precise application.
	2	Enhanced agricultural productivity through automated swarm tasks.	- Percentage increase in agricultural output. - Number of automated tasks successfully completed by swarm robots. - Reduction in labor costs associated with agricultural processes.
(10) Reduced Inequality	1	Deployment of swarm robots in difficult-to-access environments to deliver essential services like food and medicine.	- Number of successful deliveries made by swarm robots in remote areas. - Reduction in delivery time for essential services. - Increase in the number of underserved populations receiving aid.
	2	Improved access to services for underserved populations.	- Percentage of underserved populations reached by swarm robots. - Improvement in health and well-being metrics in areas served by swarm robots. - Decrease in the disparity of service access between urban and rural areas.

Table 1.1: Contributions of the Dissertation to Sustainable Development Goals.

1.4 Dissertation Outline

Excluding the Introduction chapter, the remainder of the dissertation in structured as follows:

- Chapter 2 provides a comprehensive overview of the necessary theoretical background, including the Basics of Cryptography and Homomorphic Encryption Principles, with a focus on Paillier's Encryption scheme. Additionally, it analyzes the existing literature on classical

[1] https://sdgs.un.org/goals

methods for motion control of swarm agents, the current state-of-the-art in applying homomorphic encryption to motion control, and identifies gaps and opportunities for further research.

- Chapter 3 delineates the system architecture, outlining the structure of the project divided into the environment and agent components.
- Chapter 4 provides an initial analysis of motion control of a swarm of agents without Homomorphic Encryption (HE), focusing on the control laws with and without object detection.
- Chapter 5 details the adaptation of the previously studied control laws to suit Paillier's encryption method, implementing motion control of a swarm of agents with Homomorphic Encryption.
- Chapter 6 explores the application of motion control with Homomorphic Encryption to multiple families of robots, adapting the studied methods to accommodate a scalable number of robots.
- Chapter 7 analyzes potential system improvements to mitigate the primary drawback of data encryption: the extensive computational time. This chapter explores optimization techniques and alternative approaches to enhance the efficiency of the encrypted motion control system.
- Chapter 8 concludes the dissertation by summarizing the findings and discussing potential directions for future research.

Chapter 2

Foundations and Advances in Encrypted Swarm Coordination

2.1 Basics of Cryptography

2.1.1 Introduction

Cryptography is the discipline dedicated to securing communications by transforming plaintext into ciphertext through mathematical algorithms, ensuring that adversaries cannot decipher or access the information contained within the ciphertext [13]. In light of this, a *cryptosystem* is a "set of cryptographic algorithms required to implement any security service" [13]. It consists of three algorithms: **key generation**, **encryption** and **decryption**. The components integral to its the functioning include:

- **Plaintext**: the original, unencrypted message that needs to be secured;
- **Encryption algorithm**: the mathematical procedure used to convert plaintext into ciphertext;
- **Ciphertext**: the encrypted message the encryption algorithm produces.
- **Decryption algorithm**: the mathematical procedure used to convert ciphertext back into plaintext;
- **Encryption key**: the key used in the encryption algorithm to encrypt the plaintext;
- **Decryption key**: the secret key used in the decryption algorithm to decrypt the ciphertext.

Figure 2.1 provides a schematic representation of a basic encryption system.

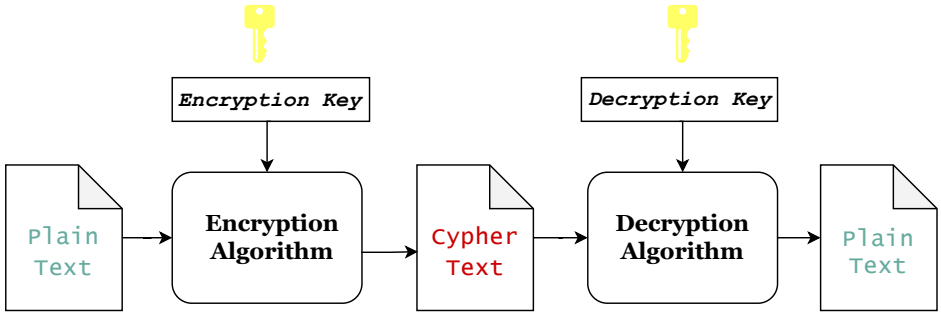


Figure 2.1: A basic encryption system.

2.1.2 Cryptography Goals

As it is explained in [16], cryptography encompasses a set of techniques designed to fulfill four fundamental objectives:

- **Confidentiality**, employed to restrict access to information content, only to those who are permitted;
- **Data integrity** is a principle that prevents unauthorized changes to data, ensuring the ability to spot unauthorized alterations, such as addition, removal, or substitution of it.
- **Authentication** is a process associated with *verification*, applicable to individuals and the information itself: information transmitted over a channel should be verified in terms of its origin, date of creation, data content, time of transmission, etc.
- **Non-repudiation** is a service that prevents an entity from denying past commitments or actions. In situations where conflicts occur due to an entity's denial of certain actions, a resolution method is necessary, for instance, one entity might approve another entity's property purchase and later deny giving such approval. A system involving a trustworthy third party is needed to resolve such disputes.

Cryptography's fundamental goal is to address these four areas in both theory and practice: it is about the prevention as well as the detection of cheating and other malicious activities [16].

2.1.3 Key Encryption

Secret Key or Symmetric Key Encryption

Secret key encryption, also known as symmetric key encryption, is a fundamental concept in cryptography where a single key is used for both encryption and decryption [13], as illustrated in Figure 2.2. This mechanism means the same key transforms plaintext into ciphertext and back into plaintext. The keys involved can either be identical or have a straightforward transformation between them, ensuring the integrity of the encryption-decryption process [17].

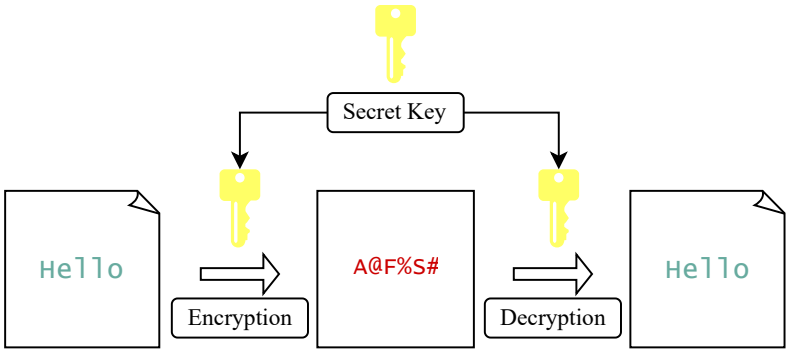


Figure 2.2: Secret key encryption model.

In real-world applications, these keys function as a shared secret between two or more individuals, enabling them to create a private channel of communication. Protecting the integrity and secrecy of the information communicated is greatly aided by this shared secret [17]. Nonetheless, safely distributing the key is a serious obstacle to symmetric key encryption. Transmitting the key over a secure channel becomes troublesome since it must be accessible to all parties involved [13]. Given that maintaining the confidentiality of the key is essential to the overall security of the system, this key distribution problem is a significant disadvantage of symmetric key cryptography. Ensuring the key remains secret during transmission is essential, yet this remains one of the most difficult aspects to manage effectively.

Public Key or Asymmetric Key Encryption

Asymmetric key encryption, or public key encryption, is a cryptographic technique that uses a public key and a private key, which are two separate keys. Though they serve different purposes, these keys are connected mathematically. Plaintext is encrypted using the public key, while ciphertext is decrypted using the private key. Unlike symmetric key encryption, which uses the same key for both procedures, "asymmetric" key encryption uses separate keys for encryption and decryption [17].

Every user has a distinct set of keys when using asymmetric key encryption: a public key for encryption and a private key for decryption. While the private key is kept secret, the public key is shared widely [13], depicted in Figure 2.3. Notwithstanding their mathematical relationship, the system is secure because it is computationally impractical to obtain the private key from the public key [13].

The process of sending encrypted data to a recipient involves retrieving the recipient's public key from the repository, encrypting the material, and then sending it. Subsequently, the recipient decrypts the data using their private key [13]. Using this method ensures that the encrypted data cannot be decrypted without the matching private key, even if it is intercepted.

Asymmetric key encryption's comparatively slower performance when compared to symmetric key encryption is one of its major drawbacks. This slower pace is a result of the encryption and

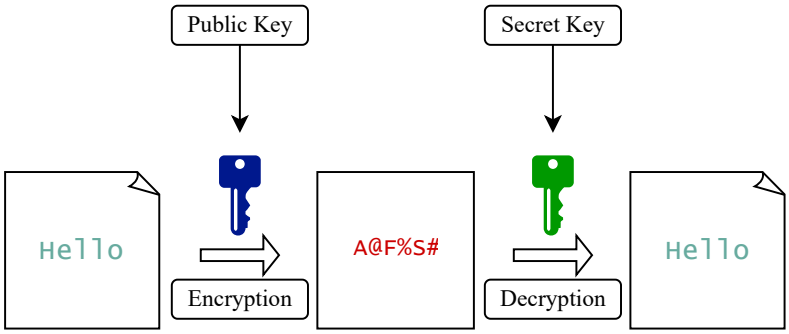


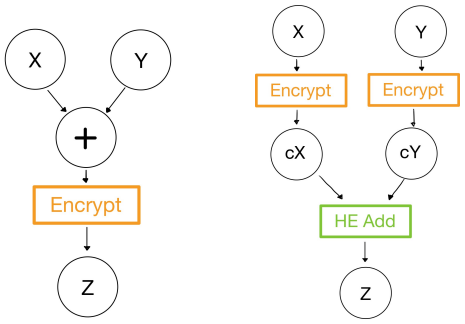
Figure 2.3: Public key encryption model.

decryption operations' greater key lengths and more complex algorithms [13]. Asymmetric key encryption is nevertheless an essential part of contemporary cryptographic systems because of its improved security and advantages in key management.

2.2 Homomorphic Encryption Principles

2.2.1 Definition

Homomorphic encryption stands as a form of encryption that allows computations to be conducted directly on encrypted data, eliminating the need for decryption and preserving the confidentiality of sensitive information throughout the control process [18]. This intuitively entails that if two plaintexts are considered, X and Y, as depicted in Figure 2.4a, along with their corresponding ciphertexts, cX and cY, presented in Figure 2.4b, operations such as addition or multiplication applied to X and Y mirror analogous operations executed on cX and cY.



(a) Without HE. (b) With HE.

Figure 2.4: Homomorphic Encryption logic.

As in abstract algebra, "a homomorphism is a structure-preserving map between two algebraic structures, such as groups" [17], and a *group* is merely a set G equipped with a single binary operation $\circ$ that combines any two elements of the set to form another element within the same

set. The set G and the operation $\circ$ must satisfy four fundamental properties, known as the group axioms [17]:

1. **Closure**: For all elements $a, b \in G$, the result of the operation $a \circ b$ is also in G.
2. **Associativity**: For all elements $a, b, c \in G$, the equation $(a \circ b) \circ c = a \circ (b \circ c)$ holds.
3. **Identity Element**: There exists an element $e \in G$ such that for every element $a \in G$, the equation $e \circ a = a \circ e = a$ is satisfied.
4. **Inverse Element**: For each element $a \in G$, there exists an element $b \in G$ such that $a \circ b = b \circ a = e$, where e is the identity element.

These properties ensure that a group is a structured and well-behaved set under the defined operation.

Based on these premises, a few points are to be noted:

1. The identity element of a group G is often written as 1;
2. The result of an operation can depend on the order of the operands. Specifically, the outcome of combining element a with element b may differ from that of combining element b with element a; hence, the equation $a \circ b = b \circ a$ does not necessarily hold in general. However, this commutative property is always true in the group of integers under addition, as $a + b = b + a$ for any two integers, demonstrating the commutativity of addition. Groups in which the commutative property $a \circ b = b \circ a$ consistently holds are termed *abelian* groups.
3. Given two groups $(G, \circ)$ and $(H, *)$, a group homomorphism from $(G, \circ)$ to $(H, *)$ is a function $f : G \to H$ such that for all g and g' in G, the function satisfies $f(g \circ g') = f(g) * f(g')$ [17] (as it is illustrated in figure 2.5).

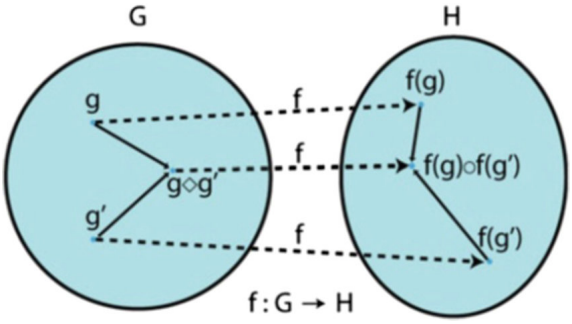


Figure 2.5: Group Homomorphism [17].

In light of the preceding, let us examine an encryption scheme such as (P, C, K, E, D) (with P as the plaintext, C as the cyphertext, K as the key space and E, D are the encryption and decryption algorithms, respectively), where the plaintexts form a group, $(P, \circ)$ as well as the cyphertexts $(C, *)$ (as stated in [17]). It can then be inferred that the encryption algorithm E is a map from the group

P to the cryptosystem C like $E_k : P \rightarrow C$, with $k \in K$ is a secret key (in a secret key cryptosystem) or a public key (in a public-key cryptosystem).

For all a and b in P and k in K, if

$$E_k(a) \circ E_k(b) = E_k(a * b) \tag{2.1}$$

then the encryption scheme is **homomorphic**.

2.2.2 Types of Homomorphic Encryption

Building upon the foundational principles of homomorphic encryption, exploring the various types of encryption schemes is essential. These types differ in their capabilities and applications, catering to specific needs and use cases within the broader landscape of secure computation.

Partially Homomorphic Encryption

Partially Homomorphic Encryption (PHE) schemes support only a single type of operation (either addition or multiplication, but not both) on ciphertexts [4]. An exemplary scheme that supports additive homomorphism is the Paillier cryptosystem, where the product of two ciphertexts decrypts to the sum of their corresponding plaintexts. Conversely, the RSA cryptosystem can be used to illustrate a multiplicative homomorphic encryption scheme, where the multiplication of two ciphertexts results in a ciphertext that decrypts to the product of the original plaintexts [4].

PHE is useful in scenarios where only one type of arithmetic operation is required. For instance, in the context of swarm robotics, if the operation required is solely the aggregation of sensor data from multiple robots, an additively homomorphic encryption scheme like Paillier would suffice. This limited functionality, however, restricts the broader application of PHE in complex robotic computations where both addition and multiplication might be required.

Somewhat Homomorphic Encryption

Somewhat Homomorphic Encryption (SHE) schemes extend the capability of PHE by supporting both addition and multiplication operations, but only up to a certain complexity or depth of computation. The limitation on the depth of computation is due to the noise that accumulates with each operation. If the noise grows beyond a certain threshold, the ciphertext can no longer be correctly decrypted [4].

A well-known SHE scheme is the Boneh-Goh-Nissim (BGN) cryptosystem, which allows an arbitrary number of additions but only one multiplication. In swarm robotics, SHE can be leveraged for more complex tasks than PHE. For example, in distributed decision-making algorithms, where a combination of summation and product operations on encrypted data from various robots is necessary, SHE schemes provide a middle-ground solution. However, the inherent noise limitations necessitate careful planning and optimization of the computation depth to ensure the correctness of the final output.

Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is the most powerful form of homomorphic encryption, allowing unlimited additions and multiplications on ciphertexts. This capability enables arbitrary computations on encrypted data, making FHE highly valuable for secure data processing in various sensitive applications. A key feature of FHE is the bootstrapping technique, which refreshes noisy ciphertexts by performing homomorphic decryption and re-encryption, effectively reducing the noise and allowing indefinite computations without compromising decryption reliability[4].

FHE provides unparalleled flexibility and security for complex, distributed computations. For instance, in scenarios where multiple entities must collaborate on encrypted data without revealing their individual inputs, FHE ensures that the computations can be performed securely and accurately. This capability is essential for robust privacy and confidentiality applications, such as secure voting systems, confidential data analysis, and private machine learning.

Despite its powerful capabilities, the practical deployment of FHE is currently limited by significant computational overhead, making it computationally intensive and resource-demanding. This can be a constraint in real-time applications where timely decision-making is critical. Current research focuses on optimizing FHE schemes to reduce computational costs and improve efficiency, aiming to make FHE more practical for real-world applications.

2.2.3 Remarks and Discussion

Naturally, selecting an encryption scheme was a pivotal aspect of this research, and the Paillier's encryption scheme was identified as the most suitable choice. This decision was predicated on several distinctive characteristics of the Paillier's encryption scheme that are particularly advantageous for the application of swarm robotics.

One of the main advantages of the Paillier's encryption scheme is its simplicity: the encryption and decryption processes are relatively straightforward. Furthermore, it is semantically secure, meaning that it is computationally infeasible to derive meaningful information from the encrypted data without the decryption key. This feature is crucial in ensuring the confidentiality of the data in our swarm robotics application.

Lastly, the public-key nature of the Paillier's encryption scheme allows for secure communication between the robots in the swarm. Each robot can have its own pair of public and private keys, allowing it to encrypt data that can only be decrypted by a robot with the corresponding private key. This ensures the integrity and authenticity of the data being communicated within the swarm.

In conclusion, the choice of the Paillier's encryption scheme was primarily driven by its homomorphic properties, simplicity, semantic security, and public-key nature. These characteristics align well with the requirements of our swarm robotics application and the author's initial lack of background in encryption.

2.3 Paillier Encryption Scheme

This section explores the Paillier encryption scheme, covering its key generation process, encryption and decryption methods, homomorphic properties, and security considerations [17] [22].

2.3.1 Key Generation

The key generation process in the Paillier encryption scheme involves the following steps:

1. **Prime Number Selection:** Two large prime numbers, denoted as p and q, are chosen randomly and independently of each other, such that

$$gcd(pq,(p-1)(q-1)) = 1 \tag{2.2}$$

   where *gcd* stands as *greatest common divisor*.

   This property is assured if both primes are of equal length.

2. **Modulus Calculation:** The public key modulus, denoted as n, is computed as the product of p and q, i.e., $n = p \times q$. A secondary modulus, n^2, is also computed and used in the encryption and decryption processes.

3. λ **Calculation:** The Carmichael's function, denoted as $\lambda(n)$, is calculated. This function is defined as the least common multiple of $p-1$ and $q-1$, i.e.,

$$\lambda(n) = lcm(p-1,q-1) \tag{2.3}$$

   This value is kept private and used in the decryption process.

4. **Public Key Generation:** The public key is represented as a pair (n,g), where n is the modulus calculated earlier, and g is a random integer such that $g \in Z^*_{n^2}$. $Z^*_{n^2}$ denotes the set of integers modulo n^2 that are coprime to n^2. In other words, $g \in Z^*_{n^2}$ means g is an integer such that $\gcd(g,n^2) = 1$. This set forms a group under multiplication modulo n^2, containing all integers between 1 and n^2-1 that share no common factors with n^2.

5. μ **Calculation:** Confirm that n is a divisor of the order of g by verifying the existence of the following:

$$\mu = (L(g^{\lambda} \mod n^2))^{-1} \mod n \tag{2.4}$$

   where L is defined as

$$L(u) = \frac{u-1}{n} \tag{2.5}$$

6. **Private Key Generation:** The private key is represented as a pair (λ,μ).

The public key is used for the encryption process, while the private key is used for decryption. The security of the Paillier encryption scheme relies on the difficulty of factoring the modulus n,

Algorithm 1: Key Generation for Paillier Cryptosystem

Input: Security parameter k
Output: Public key (n, g), Private key (λ, μ)

Selection of two large prime numbers p and q of equal length such that $gcd(pq, (p-1)(q-1)) = 1$ is true;
Computation of $n = pq$ and $\lambda = \text{lcm}(p-1, q-1)$;
Selection of $g \in \mathbb{Z}^*_{n^2}$;
Computation of $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$;
return *Public key:* (n, g), *Private key:* (λ, μ)
Function `L(u)` **:**
    Input: $u \in \mathbb{Z}^*_{n^2}$
    return $(u-1)/n$;

which is the product of two large prime numbers. This makes the key generation process a critical component of the Paillier encryption scheme.

The full algorithm is summarized in algorithm 1.

2.3.2 Encryption and Decryption

Encryption

1. **Message Selection:** Let m be the message that is to be encrypted, where m is an element of the set of integers modulo n.
2. **Random Number Selection:** Choose a random number r from the multiplicative group of integers modulo n.
3. **Ciphertext Calculation:** The ciphertext c is computed using the equation $c = g^m \cdot r \mod n$, where g is a predetermined element from the same group.

Decryption

1. **Ciphertext Selection:** Let c be the ciphertext that is to be decrypted, where c is an element of the set of integers modulo n^2.
2. **Plaintext Calculation:** The plaintext message m is computed using the equation $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$, where L and μ are operations applied within the context of the decryption process.

2.3.3 Practical Example

In this subsection, a practical example is provided to illustrate the steps involved in the key generation, encryption, and decryption processes of the Paillier encryption scheme [14].

Key Generation Example

1. **Prime Number Selection:**

   - Select two large prime numbers. To maintain computational simplicity while illustrating the concept, we shall opt for smaller primes, which, it should be noted, would not be selected in practical applications due to security concerns, $p = 3$ and $q = 5$.

2. **Modulus Calculation:**

   - Compute n as the product of p and q:

   $$n = p \times q = 3 \times 5 = 15$$

   - Compute the secondary modulus n^2:

   $$n^2 = 15^2 = 225$$

3. λ **Calculation:**

   - Calculate the Carmichael's function $\lambda(n)$, which is the least common multiple (LCM) of $p-1$ and $q-1$:

   $$\lambda(n) = \text{lcm}(2,4) = 4$$

4. μ **Calculation:**

   - Choose a random integer $g \in \mathbb{Z}^*_{n^2}$. Let's assume $g = 4$.
   - Calculate $L(g^\lambda \mod n^2))$, knowing $L(u) = \frac{u-1}{n}$:

   $$g^\lambda \mod n^2 = 4^4 \mod 225 = 31$$

   $$L(31) = \frac{31-1}{15} = 2$$

   - Calculate the multiplicative inverse of $L(g^\lambda \mod n^2)$.

     Note: To determine the multiplicative inverse of a number a modulo n, one must find a number b such that:

     $$a \times b \equiv 1 \pmod{n}$$

     This requires identifying b for which the product $a \times b$ results in a remainder of 1 when divided by n. In the given example, with $n = 15$, we need to find b such that $2 \times b \equiv 1 \pmod{15}$. By calculation, it is evident that $2 \times 8 = 16$, and since $16 \pmod{15} = 1$, $b = 8$ is the multiplicative inverse of 2.

     Therefore, we conclude:

     $$\mu = 8$$

5. **Public and Private Key Generation:**
   - Public key: $(n, g) = (15, 4)$
   - Private key: $(\lambda, \mu) = (4, 8)$

Encryption Example

1. **Message Selection:**
   - Select a message m, where $m \in \mathbb{Z}_n$. Assume $m = 4$.
2. **Random Number Selection:**
   - Choose a random number $r \in \mathbb{Z}_n^*$. Assume $r = 1$.
3. **Ciphertext Calculation:**
   - Compute the ciphertext c:

$$\begin{aligned} c &= g^m \cdot r^n \pmod{n^2} \\ &= 4^4 \cdot 1^{15} \pmod{225} \\ &= 31 \end{aligned}$$

Decryption Example

1. **Ciphertext Selection:**
   - Let $c = 31$ be the ciphertext.
2. **Plaintext Calculation:**
   - Compute the plaintext message m:

$$m = L\left(c^\lambda \pmod{n^2}\right) \cdot \mu \pmod{n}$$

Calculate:

$$\begin{aligned} L\left(c^\lambda \pmod{n^2}\right) &= \frac{31^4 \pmod{225} - 1}{15} \\ &= \frac{120}{15} \\ &= 8 \end{aligned}$$

$$\begin{aligned} m &= 8 \cdot 8 \pmod{15} \\ &= 64 \pmod{15} \\ &= 4 \end{aligned}$$

The result is $m = 4$, confirming the original message.

From key generation through encryption and decryption, this real-world example demonstrates the entire cycle of the Paillier encryption system.

2.3.4 Homomorphic properties

The Paillier encryption scheme's homomorphic characteristics are an essential feature that makes it more useful in contemporary cryptographic applications. The mathematical underpinnings of the Paillier cryptosystem's support for scalar multiplication and additive homomorphism are thoroughly examined and explained in this section.

Homomorphic Addition of Plaintexts

Formally, if $E(\ m_1, p_k)\ = g^{m_1}\ r_1^n \pmod{n^2}$ and $E(\ m_2, p_k)\ = g^{m_2}\ r_2^n \pmod{n^2}$, with random values r_1 and r_2, respectively, then the product of these ciphertexts will decrypt to the sum of the plaintexts [17], i.e.,

$$D(E(m_1, p_k) \cdot E(m_2, p_k) \pmod{n^2})\ =\ m_1 + m_2 \pmod{n} \tag{2.6}$$

That is because

$$\begin{aligned} E(m_1, p_k) \cdot E(m_2, p_k)\ &=\ (g_1^{m_1}\ r_1^n)(g_2^{m_2}\ r_2^n) \pmod{n^2} \\ &=\ g^{m_1+m_2}\ (r_1 r_2)^n \pmod{n^2} \\ &=\ E(m_1 + m_2,\ p_k) \end{aligned}$$

For the same reason, a ciphertext that is multiplied by a plaintext raising g will decode to the total of the associated plaintexts, that is,

$$D(E(m_1, p_k) \cdot g^{m_2}) \pmod{n^2})\ =\ m_1 + m_2 \pmod{n} \tag{2.7}$$

because,

$$\begin{aligned} E(m_1, p_k) \cdot g^{m_2}) &= (g_1^{m_1}\ r_1^n)\ g_2^{m_2} \pmod{n^2} \\ &=\ g^{m_1+m_2}\ r_1^n \pmod{n^2} \\ &=\ E(m_1 + m_2,\ p_k) \end{aligned}$$

Homomorphic Multiplication of Plaintexts

Conventionally, raising an encrypted plaintext to the power of another plaintext will cause it to decrypt into the sum of the two plaintexts, meaning,

$$D(E(m_1, p_k)^{m_2} \pmod{n^2}) \;=\; m_1\, m_2 \pmod{n} \tag{2.8}$$

by reason of,

$$\begin{aligned} E(m_1, p_k)^{m_2} &= ((g_1^{m_1}\; r_1^n))^{m_2} \pmod{n^2} \\ &= g^{m_1+m_2}\; (r_1^{m_2})^n \pmod{n^2} \\ &= E(m_1 \cdot m_2,\; p_k) \end{aligned}$$

In a broader context, when an encrypted plaintext is raised to a constant k, the resulting decryption yields the product of the original plaintext and the constant, i.e.,

$$D(E(m_1, p_k)^k \pmod{n^2}) \;=\; k\, m_2 \pmod{n} \tag{2.9}$$

Nevertheless, considering the two messages' Paillier encryptions, it is impossible to calculate the encryption of the message product without the private key. [17]

2.4 Swarm Coordination Robotic Systems

Swarm robotics, as described by [19], refers "to the coordination of multiple individual entities, which traditionally operate without centralized control and instead rely on simple local behaviors to cooperate." Inspired by the collective behavior observed in nature, such as flocks of birds or colonies of ants, the field focuses on planning, constructing, and utilizing large groups of robots that collaborate to accomplish tasks or solve problems [8].

Its applications span various domains, including terrestrial, aquatic, and even outer space environments, making swarm robotics an increasingly researched and studied area [8] [19].

On land, one of the many uses of swarm robots is to help with agriculture [19]. In [5], the authors aim to leverage the flexibility, scalability, and robustness of swarm robotics to improve efficiency and system robustness in agricultural tasks, particularly weed control. The study presents a road map for moving from laboratory research to practical field applications, with a particular emphasis on the SAGA project [1], which aims to use a swarm of UAVs to monitor and map weeds in agricultural fields. This strategy aims to cut labor and operational expenses while maximizing production and decreasing chemical usage, highlighting the potential of swarm robots in solving difficult agricultural challenges in a decentralized and scalable manner.

Through the aerial landscape, swarms of robots, particularly drones, are increasingly utilized in various applications such as *security and surveillance* [1]. These multi-agent systems offer

[1] https://projectsaga.eu/

significant advantages over single-agent systems by covering larger areas in shorter periods, enhancing the efficiency and effectiveness of monitoring tasks. Another notable application is *environmental monitoring* [1], where swarms of drones can form a multi-UAV system for real-time flood monitoring and tracking. This approach provides a higher accuracy and responsiveness compared to conventional forecasting methods, allowing for more precise and timely interventions in flood-prone areas [2].

Deep-space exploration missions present a great deal of risk and expense, along with a great deal of unpredictability in the operating environment. The implementation of swarm robots can reduce the consequences of failures in individual robots by facilitating cooperative job completion. The authors of [12] examine a situation in which human scientists delegate tasks to a multi-robot system that is allowed to explore Mars autonomously. Each robot uses an experience sample optimizer and a multi-agent deep deterministic policy gradient algorithm (MADDPG) to determine the best policies. The outcomes indicate that this method outperforms conventional deep reinforcement learning methods, particularly when dealing with an increasing number of robots and exploration objects.

Swarm robots can also be utilized in military operations, functioning effectively in environments too dangerous or inaccessible for humans, as well as in disaster relief efforts and healthcare settings [11].

However, achieving swarm behavior in robots goes beyond merely applying swarm intelligence algorithms to existing robotic platforms. In reality, it often necessitates a complete reevaluation of traditional robotic activities, including perception, control, localization, and even the fundamental design of the robotic platforms themselves [8]. Beyond current limitations in robot hardware and control [8], the main issue that keeps swarm agents from becoming widespread in critical areas is the associated security problem, as to be able for each robot to coordinate its action with another, one must know critical information of the other [11]. This requires the creation of a safe conduct connecting both parties, which, in this dissertation, is the application of encryption methods to "mask" the real - crucial - information.

2.5 Security Challenges

Cyberattacks on control systems fall into one of two categories: *active* or *passive*.[20]

Active Attacks involve deliberate and intrusive actions executed by threat actors to manipulate, disrupt, or compromise the targeted system. They are known to endanger the integrity and availability of data and are deemed more hazardous than passive attacks due to their direct interaction with the target system or network. Examples of such are data injection or spoofing. Nonetheless, active attacks can be largely prevented by implementing conventional cryptographic authentication protocols on transmitted data or employing anomaly detection systems [20].

Passive Attacks, on the other hand, involve an adversary observing or duplicating communication content: they primarily extract data from the system, posing a significant threat to the confidentiality of data (for example, eavesdropping) [20].

2.6 Integration of Encryption in Swarm Systems

The integration of encryption into swarm systems, as previously stated, is of utmost importance for securing communication and data exchange among agents: in a decentralized network where numerous agents interact and share information, ensuring the confidentiality, integrity, and authenticity of data is paramount. In light of this, encryption technologies provide the necessary framework to safeguard against eavesdropping, tampering, and unauthorized access [21].

There has been some research dedicated to improving security in swarm coordination, most of which involves the use of hash chains [10] [7]. *Hash Chains*, in the context of cryptography, is a sequence of hash values derived from an initial value through the iterative application of a hash function. The process begins with an initial value x_0, and each subsequent value x_i is generated by hashing the previous value, i.e.$x_i = H(x_{i-1})$, where H is a cryptographic hash function. This chain of hashes ensures that each value is computationally difficult to reverse, preserving the integrity and security of the sequence.

However, despite their widespread utility in computing, hash chains pose notable disadvantages that can hinder their effectiveness, such as their inability to permit null values - limiting their applicability in certain contexts - their finite capacity - ultimately requiring complex rehashing techniques when they become full - or their inherently intricate implementation, demanding careful design to achieve optimal efficiency.

To address these drawbacks, the point of the research changes towards homomorphic encryption, specifically Paillier's encryption scheme. Unlike hash functions, Paillier encryption does not suffer from null value restrictions or finite capacity issues. Its homomorphic properties allow for the secure aggregation and computation of data across multiple agents, ensuring that the overall system can perform complex tasks while keeping sensitive information confidential.

2.6.1 Practical Application of Paillier Encryption in Swarm Systems

In a notable study addressing the need for privacy-preserving coordination among multiple robots [15], researchers explored two key problems: privacy-preserving rendezvous and persistent monitoring. In the privacy-preserving rendezvous scenario, robots collaboratively develop a plan to meet without any robot knowing the precise meeting details in advance. For persistent monitoring, robots dynamically cover a region, ensuring collision-free movement while remaining unaware of each other's paths. This approach effectively inverts the traditional coordination paradigm, requiring robots to jointly determine potential path collisions without disclosing their individual routes. The study leveraged garbled circuits and homomorphic encryption to implement secure path intersection primitives, presenting both algorithms and software implementation successfully.

2.7 Future Directions

Despite significant advancements in swarm robotics and the promising integration of homomorphic encryption to enhance security, several research gaps and opportunities for further exploration

remain.

One major area needing attention is the computational overhead associated with homomorphic encryption. Although Paillier encryption offers significant security benefits, its computational complexity can be a bottleneck, particularly in resource-constrained environments. Furthermore, there is a need for standardized protocols and frameworks for the integration of homomorphic encryption in swarm robotics. Currently, different research studies use varied approaches and encryption schemes, making it difficult to compare results or develop interoperable systems. Establishing common standards and best practices would facilitate the broader adoption and scalability of secure swarm systems across various domains.

Future work should focus on developing scalable architectures for homomorphic encryption in swarm robotics. The work developed in this dissertation aims to contribute to this by proposing an architecture that can be escalated to multiple families of swarms, ensuring that diverse types of robots can securely coordinate and communicate. Further research should aim to refine and validate this architecture, exploring its application across different swarm sizes and types, from small-scale UAV swarms to large-scale terrestrial robot collectives.

Chapter 3

System Architecture

This chapter delineates the design and architectural framework of the simulation system developed to investigate the application of Paillier homomorphic encryption in the coordination of swarm agents. The study is segmented into three distinct simulation scenarios: a baseline without encryption (section 3.1), an intermediate scenario applying HE to a single family of N agents (in section 3.2), and an advanced scenario applying HE across M families of N agents each(in section 3.3).

By elucidating the system architecture in this chapter, the author aims to provide an understanding of the structural foundation that supports the implementation and evaluation of encrypted swarm coordination mechanisms in starling-inspired agents.

3.1 Baseline Architecture Without Encryption

In this section, we delve into the baseline architecture of the swarm coordination system without the application of homomorphic encryption. This foundational setup serves as a control scenario to understand the behavior and coordination mechanisms of agents in their unencrypted form. By comprehending this baseline, we establish a reference point for subsequent scenarios involving encrypted communication.

3.1.1 System Overview

The baseline architecture comprises a group of N agents.These agents are designed to follow a leader-follower paradigm, where the first agent acts as the leader, and subsequent agents follow while maintaining communication only with their immediate neighbors. The primary objective is for all agents to reach their designated end positions while navigating through obstacles.

Agent Roles and Responsibilities

- **Leader Agent (Agent 1)**: The primary agent responsible for leading the swarm. It sets the trajectory and communicates its position to the next agent.

- **Intermediate Agents (Agent 2 to Agent** $N-1$**)**: Act as bridges between the leader and the follower agents. Each intermediate agent receives positional information from its preceding neighbor and relays necessary adjustments to its subsequent neighbor.
- **Follower Agent (Agent** N**)**: The final agent in the sequence, which follows the positional data received from the preceding intermediate agent.

3.1.2 Communication Dynamics

In the unencrypted baseline, each agent utilizes a straightforward communication protocol to exchange positional data with its neighbors, detailed as follows:

- **Agent 1** communicates its current position x_1 to Agent 2. It uses a control law to adjust its position based on the difference between its current position and the desired position X_d.
- **Intermediate Agents (Agent 2 to Agent** $N-1$**)** receive the position x_{i-1} from the preceding agent and x_{i+1} from the following agent, and communicate their position x_i to the next agent. Each intermediate agent employs a control strategy that takes into account the positions of both neighboring agents.
- **Follower Agent (Agent** N**)** receives the position x_{N-1} from the preceding intermediate agent and adjusts its position based on the control law.

The intra-system dynamics for $N = 3$ are illustrated in Figure 3.1. In this figure, x_1, x_2, and x_3 represent the current positions of Agent 1, Agent 2, and Agent 3, respectively. Additionally, X_d denotes the desired destination of Agent 1 and the gains K_1, K_{21}, K_{23}, K_{32} are specified for the computation of each agent's control law, u_i, where $i = 1,\ldots,N$. These variables and, consequently, each agent's control law equations are going to be explained in more detail in subsequent chapteres (chapter 3, chapter 4 and chapter 5). At this point, the idea is to explain the communication dynamics between the agents.

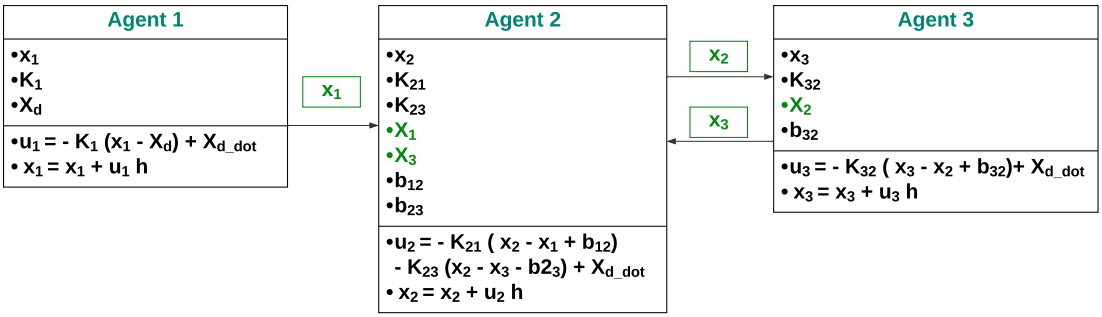


Figure 3.1: Conceptual Diagram of all Agents Motion Control.

The baseline architecture, as depicted in Figure 3.1, establishes the fundamental framework for agent communication and motion control in an unencrypted environment. This setup provides

a clear benchmark to evaluate the impact of homomorphic encryption in subsequent chapters. By understanding the dynamics and interactions in this baseline scenario, we lay the groundwork for exploring the enhancements and challenges introduced by encrypted communication in swarm coordination.

3.2 Single Family Encrypted Architecture

In this section, we introduce homomorphic encryption (HE) into the existing system of agents. The Paillier encryption scheme is applied to the communication between agents to secure the positional data exchanged during coordination. This section details the modifications to the baseline architecture to integrate HE and ensure encrypted communication while maintaining the agents' ability to reach their end positions and avoid obstacles.

3.2.1 Communication and Control Dynamics

The encrypted architecture involves the same N agents. The primary difference lies in the use of Paillier encryption to secure the data communicated between agents. Each agent encrypts its positional information before sending it to its neighbors, ensuring that sensitive data is protected during transmission.

In the encrypted architecture, the communication protocol is enhanced to include encryption and decryption steps.

- **Leader Agent (Agent 1)**: As the lead navigator responsible for steering the swarm, Agent 1 doesn't need to encrypt its position in order to compute its motion. Instead, similarly to the control law presented in 3.1, Agent 1 only needs to know its current position and the desired position X_d. Figure 3.2 illustrates just that:

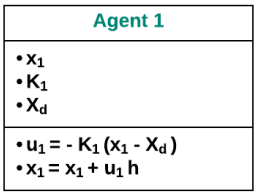


Figure 3.2: Conceptual Diagram of Agent 1.

- **Intermediate Agents (Agent 2 to Agent** $N-1$): Each intermediate agent encrypts its own position before communicating it to both its preceding and following agents, sending concurrently a public key unique to itself. Subsequently, each adjacent agent is tasked with the encryption of its respective coordinates and the computation of its contribution to the control

law of the intermediate agent, which is then relayed back. Upon receipt, the intermediate agent aggregates the incoming data and proceeds to decrypt the consolidated result utilizing its exclusive private key. The following image depicts the example of $N = 3$, which is the practical scenario implemented in this dissertation.

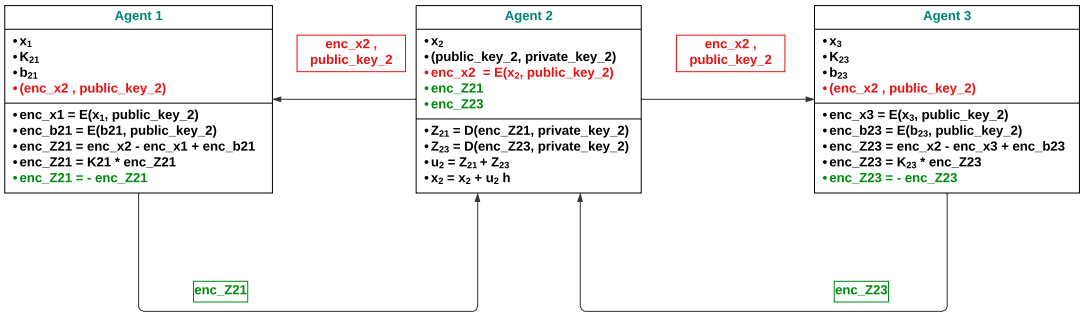


Figure 3.3: For $N = 3$, Conceptual Diagram of the Intermediate Agent.

- **Follower Agent (Agent** N**)**: Analogously to the intermediate agents, the follower agent encrypts its own position, using its characteristic public key, before sending it to the preceding intermediate agent, who is then responsible for encrypting its own position and computing its part of the control law for the follower agent. Since the follower agent is adjacent only to the preceding intermediate agent, it falls upon this preceding agent to compute the entirety of the follower agent's motion control algorithm. Upon receiving this data, the follower agent decrypts it using its individual private key.

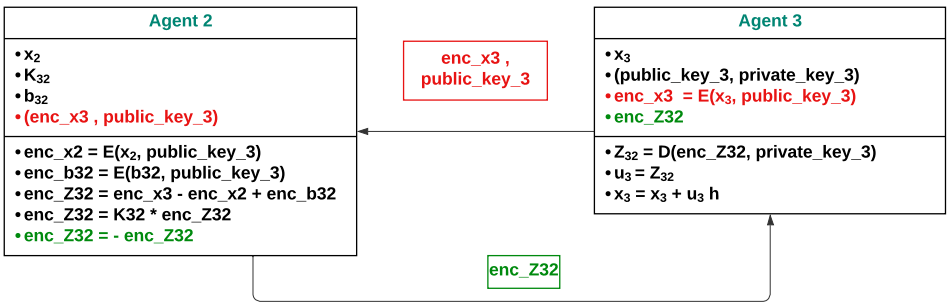


Figure 3.4: For $N = 3$, Conceptual Diagram of the Follower Agent.

The encrypted architecture, as depicted in Figures 3.2, 3.3, and 3.4, introduces Paillier encryption into the communication protocols of the agents. This setup ensures that sensitive positional data is protected during transmission, providing a secure environment for swarm coordination.

3.3 Starling-Family Encrypted Architecture

The final architecture builds upon the previously studied framework for implementing Paillier encryption, extending its application to multiple families of similar agents. This extension aims to successfully achieve the desired starling-inspired behavior. Consequently, the system now comprises N families, each containing a variable number of M agents.

3.3.1 Communication and Control Dynamics

In the Starling-Family Encrypted Architecture, the communication and control dynamics extend the principles established in the Single Family Encrypted Architecture to encompass multiple families of agents. Each family, defined as an aggregation of at least two agents, coordinates through homomorphic encryption, ensuring secure communication of positional data. Within a family, there is a mandatory Leader agent, with the remaining agents functioning as Intermediate or follower agents, collectively referred to as "Agent-Kids." In this hierarchical structure, Leaders follow other Leaders, with the notable exception of the "Head Leader" (e.g., Agent 1A), who directs the aggregations of families towards the desired location. This architecture facilitates scalable and secure swarm coordination, mirroring the intricate dynamics of starling flocking behavior.

- **"Head Leader" (Agent 1A)**: Similar to the previous depiction of the Leader Agent, in 3.2.1, Agent 1A serves as the leader agent within its family and is responsible for setting the trajectory of the . Unlike other agents, Agent 1A does not require encryption for its positional data. It calculates its position based on its control law, similar to the baseline architecture, illustrated in Figure 3.5.

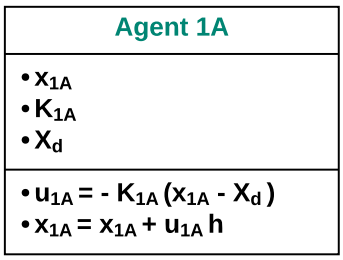


Figure 3.5: Conceptual Diagram of Agent.

- **Intermediate Agent (Agent 1B to Agent** $1(N-1)$): Intermediate Agents within each family (e.g., Agent 1B) adhere to an encryption-based communication protocol to ensure the secure exchange of positional data. Each Intermediate Agent encrypts its own positional information prior to transmitting it to adjacent agents. These agents also disseminate their unique public keys, thereby enabling the receiving agents to encrypt their positional data accordingly. Upon receipt of the encrypted data, Intermediate Agents aggregate the encrypted

contributions from neighboring agents and decrypt the consolidated result using their private keys, as detailed in Section 3.2.1. The innovation in this architecture lies in the dual role of Intermediate Agents: while they function similarly to their counterparts in the previous section, they also serve as Leader Agents for their own "Agent-Kids."

The communication dynamics for an Intermediate Agent are depicted in Figure 3.6, with an example of $N = 3$.

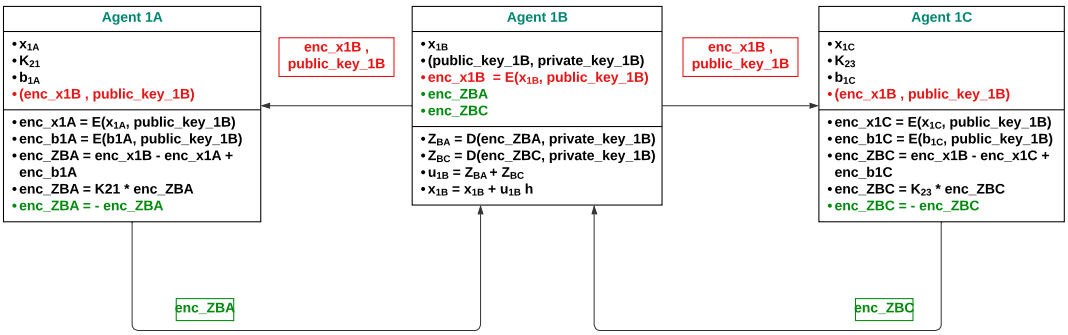


Figure 3.6: Conceptual Diagram of Agent.

- **Follower Agent (Agent** $1N$**)**: Follower Agents, like the Intermediate Agents, utilize encryption to secure their positional data during transmission. Each Follower Agent encrypts its position using its public key before sending it to the preceding Intermediate Agent. The Intermediate Agent then encrypts its position and calculates its part of the control law for the Follower Agent. The entire motion control algorithm for the Follower Agent is computed by the preceding Intermediate Agent, which then sends the encrypted data back to the Follower Agent. The Follower Agent decrypts the received data using its private key. This communication protocol is visualized in Figure 3.7

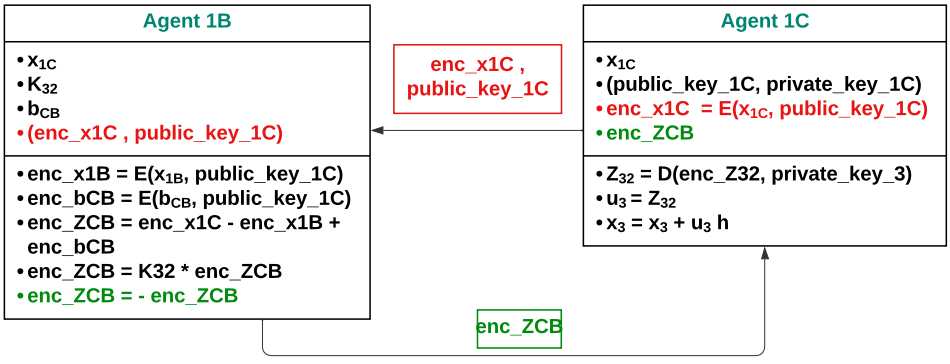


Figure 3.7: Conceptual Diagram of Agent.

3.3.2 Obstacle Avoidance Mechanism and Positional Update

Each agent within a family is equipped with local sensors to detect obstacles within its vicinity. These sensors gather data about the environment, including the location and size of potential obstacles. The positional data related to detected obstacles is not encrypted to allow for quick and efficient real-time processing and communication among agents. Once an obstacle is detected, the agents collaboratively determine an avoidance trajectory. The Leader Agent (Agent 1A) plays a pivotal role in initiating the avoidance maneuver by altering its trajectory to circumvent the obstacle. Intermediate Agents (e.g., Agent 1B) receive the positional data from the Leader Agent and adjust their paths accordingly. Although the position of the obstacle is not encrypted, the communication regarding the trajectory adjustments among agents is encrypted to maintain secure coordination.

The positional update algorithm operates as follows:

1. **Detection:** Each agent continuously scans for obstacles using local sensors.
2. **Trajectory Adjustment:** Agents collaboratively adjust their trajectories to avoid the obstacle based on local sensor data.
3. **Positional Update:** Adjusted positions are encrypted and shared within the family for immediate use by neighboring agents.

3.3.3 Considerations

The Starling-Family Encrypted Architecture leverages Paillier homomorphic encryption to secure inter-agent communication across multiple families. Each agent within a family follows a structured protocol to encrypt, communicate, and decrypt positional data, ensuring secure and coordinated movement. By extending the encryption scheme to multiple families, the system achieves scalable and secure swarm coordination, inspired by starling flocking behavior.

Chapter 4

Motion Control of a Swarm of Agents

4.1 Objectives and Challenges

This chapter covers the planning and execution of a robust control scheme for a fleet of robots assigned to obstacle avoidance and formation control. The purpose of this chapter is to demonstrate the coordinated maneuvering of the robots around obstacles, their dynamic interaction with each other and the impact of a designated leader.

By laying the groundwork in a non-encrypted context, the chapter aims to ensure:

1. **Leader-Follower Formation Maintenance**: The robots must maintain a predetermined formation with a designated leader tracking a reference trajectory;
2. **Obstacle Avoidance**: Each robot must effectively navigate around obstacles while maintaining the overall formation and avoiding collisions;
3. **Scalability**: The control strategy should be scalable to accommodate an increasing number of robots without significant loss in performance or stability;
4. **Simulate Solution**: Implement simulations that mimic swarm behavior in a controlled environment. These simulations will include scenarios where agents follow a designated leader and navigate around static obstacles, demonstrating the practical applicability of the proposed control laws.

In light of that, the development and implementation of motion control for a swarm of agents pose several inherent challenges, which need to be addressed to achieve the aforementioned goals:

1. **Limited Communication**: Each robot can only communicate with its immediate neighbors, necessitating decentralized control strategies that rely on local information;
2. **Dynamic Environments**: The robots must adapt to dynamic changes in the environment, including moving obstacles and varying trajectories;

3. **Formation Stability**: Maintaining a stable formation while navigating through an environment with obstacles requires precise coordination and control adjustments;

4. **Obstacle Avoidance Integration**: Integrating obstacle avoidance mechanisms with the formation control without causing instability or excessive deviation from the desired path is complex;

5. **Parameter Tuning**: Selecting appropriate control gains and repulsive field parameters is critical for balancing responsiveness and stability, particularly in varying environmental conditions.

In the following sections, we will explore the methodologies and strategies employed to address these challenges and meet the stated objectives, laying a strong foundation for the eventual incorporation of homomorphic encryption into swarm coordination.

4.2 Leader-Follower and Obstacle Avoidance Dynamics

This section discusses the experimental setup and procedures used to assess the performance of a fleet of robots operating in a leader-follower configuration with obstacle-avoidance capabilities. The experiment is divided into two primary parts: following a leader and obstacle avoidance for single or numerous agents.

4.2.1 Following a Leader

We consider a fleet of $n = 3$ agents (robots), each described by the following dynamics:

$$\dot{x}_i = u_i, \quad i = 1,2,\ldots n \tag{4.1}$$

where

$$x_i = \begin{bmatrix} x_{1i} \\ x_{2i} \end{bmatrix}, \quad u_i = \begin{bmatrix} u_{1i} \\ u_{2i} \end{bmatrix} \tag{4.2}$$

are the state position and control variables for each agent i, respectively.

Communication Constraints

The agents exhibit communication capabilities limited to their immediate neighbors, thereby forming a localized network for information exchange. Within this network, the leader assumes the role of agent 1, responsible for tracking a specified reference trajectory denoted as $x_d(t)$.

The feedback controller governing the leader's behavior is formally defined as follows:

$$u_1 = -K_1(x_1 - x_d) + \dot{x}_d \tag{4.3}$$

where K_1 is a 2×2 positive-definite matrix gain.

To ensure that the remaining agents follow the leader, the control input for each follower agent i follows a consensus protocol given by:

$$u_i = -K_i \sum_{j \in N_i} \big((x_i - x_j) - b_{ij}\big) + \dot{x}_d, \quad i = 2,3,\ldots n \tag{4.4}$$

where K_i is a 2×2 positive-definite matrix gain, and the bias terms b_{ij} specifies the convergence distance.

4.2.2 Obstacle Avoidance

Single Agent Scenario

For a single agent, the dynamics are defined as:

$$\dot{x} = u \tag{4.5}$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.6}$$

For illustrative purposes, we consider a scenario where the robot initiates its trajectory from the origin $[0.0, 0.0]$ and aims to reach the goal at $[1.5, 2.0]$. A circular obstacle centered at $[1.0, 1.0]$ with a radius of $r = 0.5m$ is introduced to this context, as depicted in 4.1.

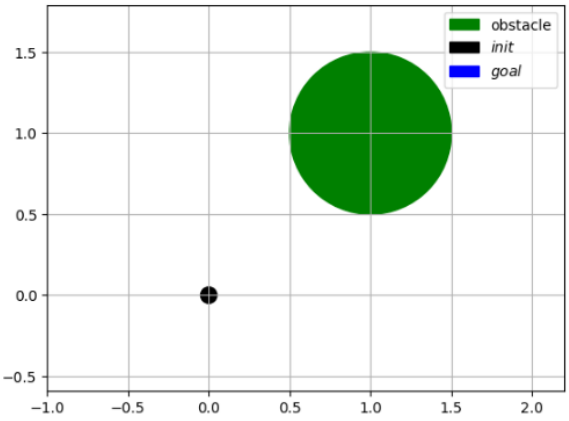


Figure 4.1: Robot, goal and obstacle.

The former controller of 4.3 is then modified to include a repulsive term to avoid the obstacle:

$$u = -K\tilde{x} + \frac{k_o(t)}{d^2}(x - x_o) + \dot{x}_d\,, \quad d = ||x - x_o|| - r \tag{4.7}$$

where $\tilde{x} = x - x_d$, $x_o = [1.0, 1.0]$ and r are the center position of the obstacle and radius, respectively, and d is the distance between the robot position and the obstacle boundary. The term $K_o(d)$ is a function that changes with the distance from the obstacle and was added in order to decrease

the influence of the repulsive term when the robot is not close to the obstacle. It varies according to:

$$k_o(d) = \bar{k}_o e^{-cd^2} \tag{4.8}$$

where $\bar{k}_o, c$ are some positive constants.

Based on the given equations, the resulting behavior should resemble the one depicted in the following illustration 4.2:

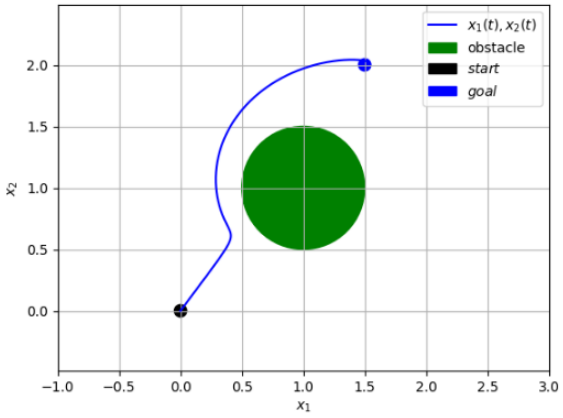


Figure 4.2: One agent expected behavior.

Multi-Agent Scenario

When extending obstacle avoidance to the entire fleet, the leader agent's control law is modified to include obstacle avoidance

$$u_1 = -K_1(x_1 - x_d) + \frac{k_{1o}(d_1)}{d_1^2}(x_1 - x_o) + \dot{x}_d \tag{4.9}$$

where $k_{1o}(d) = \bar{k}_{1o} e^{-cd_1^2}$ and $d_1 = ||x_1 - x_o|| - r$. For the agents that follow, the control law incorporates repulsive terms to avoid collisions with both the obstacle and other agents:

$$u_i = -K_i \sum_{j \in N_i} \left((x_i - x_j) - b_{ij}\right) + \sum_{j=1}^{n} \frac{k_{ij}(d_{ij})}{d_{ij}^2}(x_i - x_j) + \frac{k_{io}(d_i)}{d_{io}^2}(x_1 - x_o) + \dot{x}_d, \quad i = 2, 3, \ldots n \tag{4.10}$$

with d_{ij} as $d_{ij} = ||x_i - x_j||$ and d_{io} as $d_{io} = ||x_i - x_o|| - r$

4.3 Results

The simulations were conducted in a 2D plane where a fleet of three agents was tasked with maintaining a formation while navigating around obstacles. The initial conditions of the robots were set at different coordinates along the x-axis, with a sampling time h of 0.1 seconds and a t_{end}

of 20 seconds. The leader-follower formation and obstacle avoidance tasks were implemented as described in the methodology section.

The acquired results can be classified into three distinct aspects: trajectory tracking [4.3.1], obstacle avoidance [4.3.2], and inter-agent distance maintenance [4.3.3].

4.3.1 Trajectory Tracking

The agents start from their respective initial positions at $[-2,0]$, $[-2,2]$, $[-2,4]$ and aim to follow a designated leader moving along a reference trajectory. In this case, the leader's control input is defined to follow a desired reference x_d given by

$$x_d(t) = \begin{cases} (0,0), & 0 \leq t < 5 \\ (5,0), & t \geq 5 \end{cases}$$

with agents 2 and 3 adjusting their positions accordingly. Figures 4.3 and 4.4 show the trajectories and time evolution of the states of each agent, respectively.

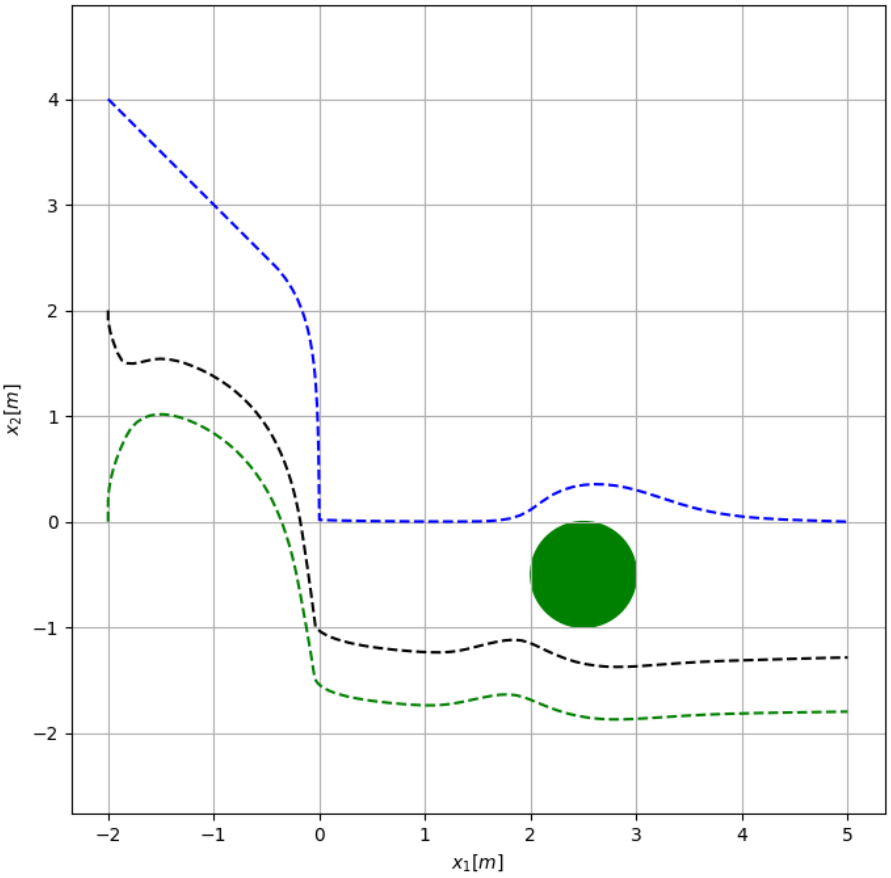


Figure 4.3: Trajectories of agents with respect to the leader's reference trajectory.

Analyzing the trajectories reveals that all agents correctly track the leader's reference trajectory while preserving formation stability. The solid lines in Figure 4.4 represent the agents' X-routes,

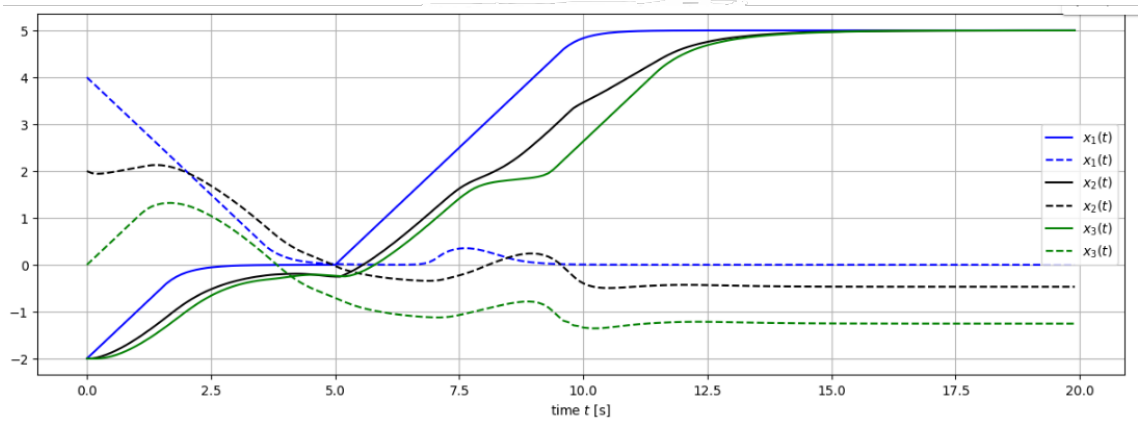


Figure 4.4: Time evolution of the state of each agent, with the *dashed* line corresponding to the Y-values of each agent, while the solid line pertains to the X-values.

and their overlap confirms the control strategy in keeping the formation's integrity, as all agents converge to the same X-coordinate, maintaining inter-agent distances denoted by b_{ij}.

4.3.2 Obstacle Avoidance

As seen in Figure 4.3, a green circular obstacle is located at $([2.5, -0.5])$ with a radius of 0.5m. Illustrated in the same figure, we observe that the agents temporarily deviate from their direct paths to circumvent the obstacle. However, they promptly resume their desired formation once the obstacle is bypassed. This demonstrates the control strategy's capability to handle obstacle avoidance without significant disruption to the formation.

4.3.3 Inter-agent Distance

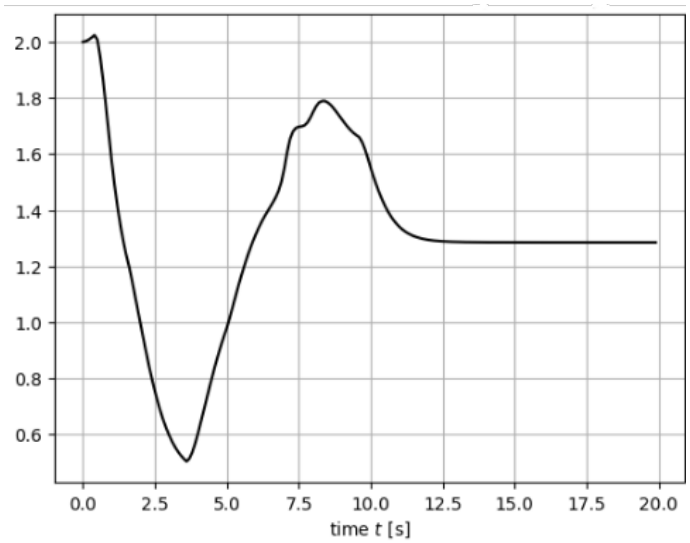


Figure 4.5: Time evolution of the distance from agent 2 to agent 1.

Figure 4.5 depicts the inter-agent distance between Agent 1 and Agent 2 over the simulation period, and from it, we observe these key behaviors:

1. **Initial Deviation and Stabilization**:
    - At the beginning of the simulation, the distance between agent 2 and agent 1 starts at 2.0 m.
    - The distance rapidly decreases, indicating that agent 2 is moving towards agent 1 to establish the formation. This is expected as the agents adjust their initial positions to form the desired leader-follower structure.
2. **Fluctuations and Adjustment**:
    - Between 2s and 10s, there are noticeable fluctuations in the inter-agent distance. This period is characterized by the agents fine-tuning their positions relative to each other and the obstacle.
    - The Agent's trajectory diverges with a peak around 8s at 1.8m before stabilizing. These fluctuations can be attributed to the dynamic adjustments required for obstacle avoidance. This part of the simulation is more noticeable in the video animation, where the differences in speed and responsiveness between the agents become evident.[1]
3. **Convergence to Desired Distance**:
    - After 12s, the distance between agent 2 and agent 1 stabilizes around 1.3 (value influenced by both b_{12} - which is 0.99 - and the repulsive term). This indicates that the agents have successfully converged to the desired formation distance while avoiding the obstacle.

A similar analysis can be done between Agent 3 and Agent 2, as depicted in Figure 4.6.

In the current context, it is important to underscore that the convergence distance between the two agents diverges from the previous analysis. This discrepancy stems from the distinct value assigned to the parameter (b_{32}), which stands at 0.48. Notably, this value differs from that of (b_{12}), and consequently, the disparity in parameter values directly affects the separation between the adjacent agents.

[1] Video animation available at https://drive.google.com/file/d/19m2GWewPaGXHfm4yaEnSE77-Wt9iHHAj/view?usp=sharing.

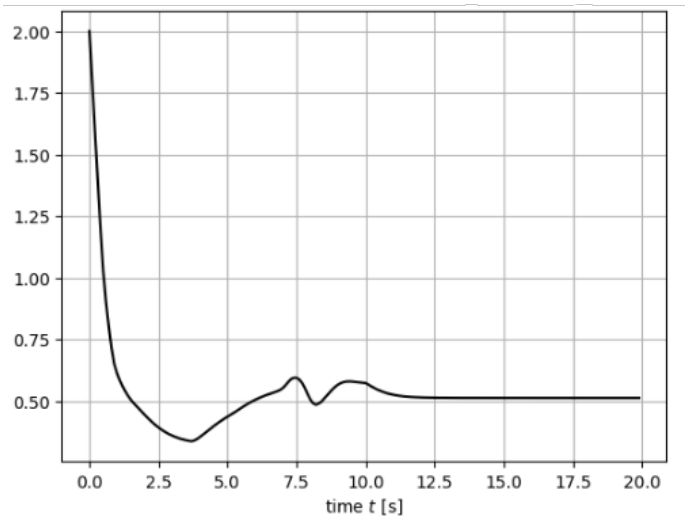


Figure 4.6: Time evolution of the distance from agent 3 to agent 2.

4.4 Considerations

In this chapter, we presented the implementation of a robust control strategy for a fleet of robots tasked with formation control and obstacle avoidance. The primary objectives encompassed maintaining a leader-follower formation, effective navigation around obstacles, ensuring scalability, and simulating the solution in a controlled environment. The key findings from our experiments demonstrated several significant outcomes.

Firstly, the robots consistently maintained a predetermined formation with the designated leader successfully tracking a reference trajectory. The control inputs for both the leader and the followers were effective, ensuring the stability of the formation even in dynamic environments. This robust leader-follower formation maintenance was a critical achievement.

Secondly, the incorporation of repulsive terms into the control laws enabled the robots to navigate around obstacles without significantly disrupting the overall formation. The simulations indicated that the control strategy effectively handled both single-agent and multi-agent scenarios, demonstrating the versatility and reliability of the obstacle avoidance mechanism.

Moreover, the control strategy exhibited impressive scalability. It accommodated an increasing number of robots without a significant loss in performance or stability, highlighting the potential for application in larger, more complex robotic systems.

Lastly, the simulations accurately reflected swarm behavior within a controlled environment. The agents not only followed the designated leader but also adeptly navigated around static obstacles. This capability underscored the practical applicability of the proposed control laws in real-world scenarios, reinforcing the validity and effectiveness of the strategy.

In summary, the experiments underscored the efficacy of the control strategy in maintaining formation, avoiding obstacles, scaling with the number of robots, and simulating swarm behavior, thereby providing a comprehensive solution for robust robotic fleet management.

.

Chapter 5

Enhancing Swarm Coordination via Homomorphic Encryption

5.1 Objectives and Challenges

The primary objective of this chapter is to successfully incorporate homomorphic encryption, specifically the Paillier cryptosystem, into the pre-existing swarm coordination system described in Chapter 4. By doing so, we aim to enhance the privacy and security of the swarm robotics system without compromising its operational efficiency or the accuracy of its collaborative tasks.

The key objectives of this chapter are outlined as follows:

- **Integrate Paillier Cryptosystem:** Implement the Paillier homomorphic encryption scheme into the existing swarm robotics framework. This involves modifying the communication protocols to ensure that positional data shared between agents is encrypted.
- **Maintain Coordination Accuracy:** Ensure that the integration of homomorphic encryption does not degrade the performance of the swarm. The system should still achieve the same level of coordination accuracy as demonstrated in Chapter 4, where each agent successfully reaches its designated end position while navigating through obstacles.
- **Enhance Security:** Increase the overall security of the swarm robotics system by protecting the positional information of each agent from potential malicious attacks. This involves ensuring that even if an agent is compromised, its exact position and the positions of its neighbors remain confidential.
- **Ensure Scalability:** Confirm that the encryption scheme can be scaled to accommodate an arbitrary number of agents in the swarm, as the system should be able to handle N agents.

Nevertheless, incorporating homomorphic encryption into a swarm robotics system presents several challenges that need to be addressed:

- **Computational Overhead:** Homomorphic encryption schemes, such as Paillier, are computationally intensive. The challenge lies in optimizing the encryption and decryption processes so that they can be performed in the minimum time possible without introducing significant delays in the system's operation.
- **Algorithm Integration:** Integrating the encryption scheme into the existing control algorithms without altering their fundamental behavior is critical. The control laws must be adapted to work with encrypted data while still achieving the desired coordination outcomes.
- **Key Management:** Efficient management of encryption keys is crucial for maintaining security. Each agent must securely generate, distribute, and manage its public and private keys to prevent unauthorized access and ensure the integrity of the encrypted communication.
- **Robustness to Attacks:** The system must be robust against various types of attacks, including eavesdropping, man-in-the-middle attacks, and physical capture of agents. This requires a comprehensive security strategy that encompasses both encryption and other protective measures.

In summary, the incorporation of the Paillier cryptosystem into the swarm robotics framework aims to achieve a balance between maintaining the system's operational efficiency and enhancing its security.

5.2 Problem Statement

In the realm of swarm robotics, maintaining the confidentiality of each robot's exact position is of paramount importance, especially in scenarios where one or more robots might fall victim to malicious attacks. Compromising the position of individual robots can lead to severe consequences, such as targeted attacks on specific robots, disruption of the swarm's coordinated behavior, and exposure of strategic information in sensitive applications. Therefore, it is essential to ensure that the positional information of each robot remains private.

The current system, as described in the previous chapter, allows each robot to share its position with its neighbors to perform collaborative tasks. However, this openness also makes the system vulnerable to security breaches. The challenge lies in balancing the need for collaboration with the necessity of maintaining position privacy. Thus, the crucial question to be addressed in this chapter is: "How can each robot's position be kept anonymous, even though, as a swarm system, each robot needs to know each neighbor's position?"

In order to tackle this issue, the proposed solution involves leveraging homomorphic encryption, more specifically, the Paillier encryption scheme, to ensure necessary calculations to be made while keeping the actual positions secret.

The expected outcomes of incorporating homomorphic encryption into the swarm robotics system include enhanced security, improved privacy of individual robot positions, and a more robust and resilient swarm coordination mechanism. By maintaining the confidentiality of positional

information, the system can continue to perform its collaborative functions effectively, even in the presence of potential security threats.

5.3 Paillier Cryptosystem Integration

In this section, the system is enhanced by encrypting the positional data exchanged between agents using the Paillier homomorphic encryption scheme. The aim is to ensure that the motion control of the swarm agents is preserved while maintaining the privacy and security of their positional data.

The current system keeps the fleet of three agents, described in the previous chapter, running in a leader-follower configuration with obstacle avoidance capabilities, each agent having the capacity to communicate with its immediate neighbors, and the leader tracking a reference trajectory.

The integration process involved the following steps:

1. **Key Generation**

   Each agent generates a pair of Paillier keys: a public key for encryption and a private key for decryption. The public keys are shared among the agents to enable encrypted communication, while the private keys remain confidential to each agent. These keys are only generated once and are an intrinsic characteristic of each Agent, meaning they never run the risk of being the same between two or more agents. The code snippet below shows the key generation for each agent, leveraging from **phe** [1] python-library:

```
from phe import paillier

public_key_1, private_key_1 = paillier.generate_paillier_keypair()
public_key_2, private_key_2 = paillier.generate_paillier_keypair()
public_key_3, private_key_3 = paillier.generate_paillier_keypair()
```

Listing 5.1: Public-Private Key Generation for each Agent

2. **Encryption of Data**

   In addition to the primary objective of protecting and therefore encrypting each agent's position during communication, another critical value requires encryption: the *bias terms.*

   The rationale behind this necessity can be elucidated by examining equation 4.10.

   This equation can be decomposed into three principal components: "maintaining formation", "collision avoidance with other agents", and "collision avoidance with obstacles". In a real-world scenario, collision avoidance with both agents and obstacles is managed by local sensors, which do not present security risks and, therefore, do not require encryption. However, the "maintaining formation" component is critical and requires careful attention.

[1] https://python-paillier.readthedocs.io/en/develop/

In the framework of this dissertation, there are three agents: Agent 1 does not require encryption, as it only adjusts its trajectory according to the desired destination; Agents 2 and 3, however, rely on their neighbors to navigate correctly. Considering Agent 2, as it is explained in 4.10, and disregarding the collision avoidance component:

$$u_2 = -K_i \sum_{j \in N_i} \big((x_i - x_j) - b_{ij}\big) = -K\big((x_2 - x_1) + b_{12} + (x_2 - x_3) - b_{23}\big) \tag{5.1}$$

As illustrated in equation 5.1, let us consider a scenario where Agent 2 encrypts its position using its public key and sends the encrypted position, along with the public key, to Agents 1 and 3. Each of these agents would then encrypt their positions and perform the subtraction$(x_i - x_j)$, where x_i is the encrypted position of Agent 2 and x_j is the encrypted position of Agent 1 or Agent 3, accordingly. They would then send the result back to Agent 2. In the event that Agent 2 behaves dishonestly, it could derive the positions of its neighbors by simply performing the inverse operations, and since the latter is the only one with the correspondent private key, it also holds exclusive access to decrypt the actual results of the operations performed.

It is therefore unfeasible to ensure security within the system, having only encrypted the actual positions of each robot. Naturally, the problem persists even if Agent 2 encrypts and sends b_{12} or $b23$.

One proposed solution to this problem is to assign specific values to the bias terms and make them intrinsic to each robot. For example, similar to how Agent 1's position is unique to itself, the bias term, b_{12}. can also be uniquely assigned to Agent 1. The same principle applies to Agent 3 and its corresponding bias term b_{23}.

The dynamics of the system then become: Agent 2 first generates its pair of public-private keys and encrypts its own position. Subsequently, it sends this encrypted position, as well as the previously generated public key, to Agents 1 and 3. Each of these agents then encrypts their own positions using Agent 2's public key, along with their unique bias terms. They then perform the calculation $(x_i - x_j) \pm b_{ij}$ and send the result back to Agent 2, who adds together both contributions. To enhance security, the proposed solution also considers making the values of K intrinsic to each robot. An advantage of this approach is that K, being a scalar parameter, does not require encryption. Thus, the actual control law of Agent 2 is as follows:

$$u_2 = -K_{21}\big((x_2 - x_1) + b_{12}\big) - K_{23}\big((x_2 - x_3) - b_{23}\big) \tag{5.2}$$

with Agent 1 performing the term $-K_{21}\big((x_2 - x_1) + b_{12}\big)$ and Agent 3 performing $-K_{23}\big((x_2 - x_3) - b_{23}\big)$.

Analogously to this reasoning, Agent 3 's control law is as follows:

$$u_3 = -K_{32}\big((x_3 - x_2) + b_{32}\big) \tag{5.3}$$

with K_{32}, b_{32} and naturally, x_2 being intrinsic values of Agent 2.

3. **Developed functions to assist in the calculations for the control law**

   `subtract_final_arrays`

   The `subtract_final_arrays` function performs subtraction on two encrypted arrays. Due to the properties of the Paillier cryptosystem, direct subtraction is not possible. Instead, subtraction is achieved by negating the second array and then performing an encrypted addition.

```
1 def subtract_final_arrays(encrypted_array1, encrypted_array2, public_key):
2
3     # Negate the second encrypted array by subtracting each element from an
          encrypted zero
4     negated_encrypted_array2 = [public_key.encrypt(0) - x for x in
          encrypted_array2]
5
6     # Perform encrypted addition (equivalent to subtraction in this context)
7     encrypted_result = [x + y for x, y in zip(encrypted_array1,
          negated_encrypted_array2)]
8
9     return encrypted_result
```

Listing 5.2: Function for subtraction

Detailed Description:

- **Input:**
  - `encrypted_array1`: The first array of encrypted values.
  - `encrypted_array2`: The second array of encrypted values to be subtracted from the first.
  - `public_key`: The public key used for encryption.
- **Process:**
  - Each element in `encrypted_array2` is negated by subtracting it from an encrypted zero. This operation uses the homomorphic property of the Paillier cryptosystem to achieve negation.
  - The resulting negated array is then added to `encrypted_array1`. Due to the homomorphic properties, this addition effectively performs the subtraction.
- **Output:**
  - An array of encrypted values representing the result of the subtraction.

`finite_addition`

The `finite_addition` function performs repeated addition of an encrypted array. This simulates scalar multiplication by adding the array to itself a specified number of times.

```
def finite_addition(encrypted_array, gain):

  aux = encrypted_array
  # Repeat the addition gain - 1 times
  for i in range(0, gain - 1):
    encrypted_array = [a + b for a, b in zip(encrypted_array, aux)]

  return encrypted_array
```

Listing 5.3: Function to perform finite addition on an encrypted array a specified number of times.

Detailed Description:

- **Input:**
  - `encrypted_array`: The array of encrypted values to be added repeatedly.
  - `gain`: The number of times the array should be added to itself.
- **Process:**
  - An auxiliary array (`aux`) is initialized to hold the original encrypted array.
  - A loop runs `gain - 1` times, in each iteration adding `aux` to the current state of `encrypted_array`.
  - This process effectively multiplies each element in the encrypted array by the scalar value `gain`.
- **Output:**
  - An array of encrypted values representing the result of the repeated addition.

4. **Decryption**

   The decryption process is the final step in each time iteration to be performed. This task is carried out exclusively by the agent to which the control law pertains and possesses the corresponding private key. The decryption function is straightforward in nature, as exemplified below:

```
def decrypt_array(encrypted_array, private_key):

    return np.array([private_key.decrypt(x) for x in encrypted_array])
```

Listing 5.4: Function for decrypting array.

Detailed Description:

- **Input:**
  - `encrypted_array`: The array of encrypted values to be decrypted.
  - `private_key`: The private key used for decryption.
- **Process:**
  - The function iterates over each element in the `encrypted_array`.
  - Each encrypted element is decrypted using the `private_key`.
  - The decrypted values are collected into a new numpy array.
- **Output:**
  - A numpy array of decrypted (plaintext) values.

5.4 Results

Just as set up in 4, the simulation was conducted in a 2D plane, with the same initial coordinates along the x-axis and the y-axis, the same 3 Agents, with a sampling time of 0.1 seconds and 20 seconds simulation time.

As anticipated, the results of the obstacle avoidance problem, when updated to include homomorphic encryption, are consistent with the outcomes presented in the previous chapter where homomorphic encryption was not employed. The result of the simulation[2] can be seen in Figure 5.1.

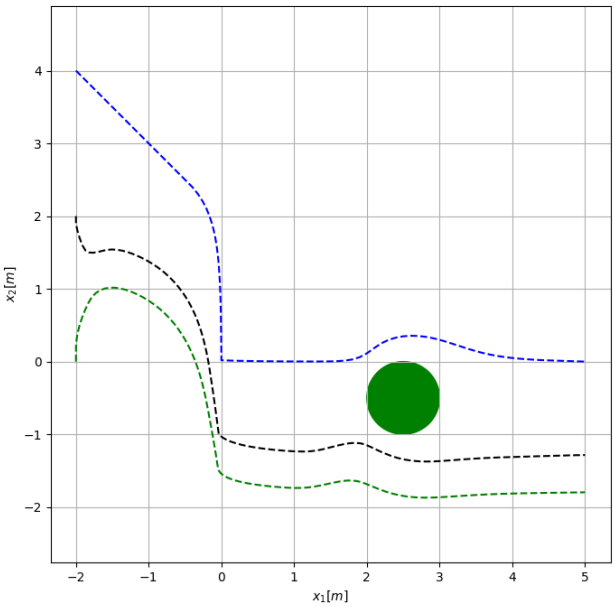


Figure 5.1: Trajectory of all 3 Agents, with the HE control implementation.

[2]Video animation in https://drive.google.com/file/d/1x1JiTVowtEyPYJBeb0aza6MbzpX_a91P/view?usp=sharing

Analogously, the time evolution of all 3 agents is also the same as depicted in 4, in Figure 4.4, confirming that the control strategy wasn't affected by the encryption scheme, as all agents kept the formation's integrity, maintaining inter-agent distances and converging all to the same X-coordinate, as illustrated by Figure 5.2,

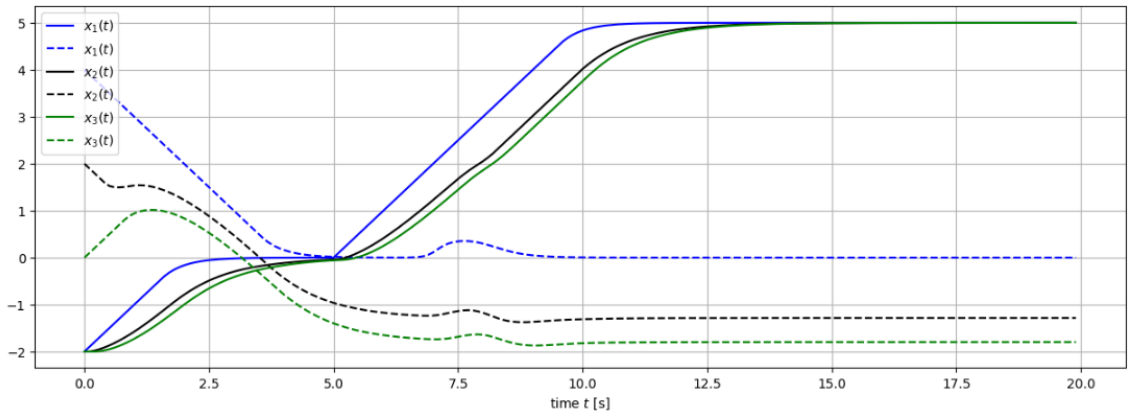


Figure 5.2: Time evolution off all 3 Agents, with HE implementation.

The outcome of this simulation stands as an achieved goal since it demonstrated that incorporating the Paillier encryption scheme doesn't hinder the accuracy of the pre-developed control law.

Despite its advantages, this implementation has a notable drawback: the substantial runtime required for full execution. Generating key pairs for each agent and encrypting the bias terms take approximately 27.5 seconds. This step is conducted separately from the actual simulation because, in a real-world scenario, it is a one-time process and not included in the overall simulation time.

However, the efficiency of the simulation itself is questionable. It takes 25.13 minutes to encrypt all agents' positions, compute their trajectories, and decrypt the results over 200 iterations. In contrast, the swarm system without homomorphic encryption completes the same simulation almost instantaneously.

5.5 Considerations

The results corroborated those presented in the previous chapter, affirming that the integration of homomorphic encryption does not compromise the accuracy of the non-encrypted control law. Consequently, the objectives outlined at the beginning of the chapter were successfully achieved, demonstrating the feasibility of this approach in maintaining theoretical accuracy. However, despite overcoming several critical challenges, such as algorithm integration, key management, and robustness to attacks, a significant issue remains unresolved: the computational overhead. This issue not only slows down the simulation process but also makes the current implementation impractical for real-world applications. The excessive runtime necessitates further optimization to ensure that the benefits of encryption do not outweigh its practical applicability, emphasizing the need for future research to address these computational constraints effectively.

Chapter 6

Motion Control with Homomorphic Encryption applied to Starling-Families

6.1 Objectives and Challenges

The primary objective of implementing motion control with homomorphic encryption in Starling-Families is to enable a swarm of agents to navigate securely and efficiently towards their designated end destinations while avoiding obstacles. This entails ensuring that each agent, equipped with local sensors and encryption capabilities, can autonomously and collaboratively make real-time decisions to maintain optimal paths and secure communication within the swarm. This section outlines the specific objectives and associated challenges inherent in achieving this goal.

- **Secure and Efficient Navigation**: Ensure that each agent in the Starling-Family can securely and efficiently navigate from its starting point to its destination while incorporating obstacle avoidance.
- **Scalability and Robustness**: Design the system to scale effectively with an increasing number of agents and families.

Nonetheless, this approach presents its own set of challenges, which include:

- **Computational Overhead of Encryption**: Homomorphic encryption offers robust security; however, it imposes substantial computational overhead. Balancing security with the requirement for real-time processing is a critical challenge.
- **Scalability of the System**:As the number of agents and families increases, managing secure communications and coordinated movements grows exponentially in complexity. Developing scalable architectures and control strategies that maintain performance and security across large swarms is vital.

All things considered, homomorphic encryption presents a number of operational and technological obstacles that must be overcome in order to achieve safe and effective mobility control in

Starling-Families. Strong, safe, and scalable swarm operations in a variety of applications can be made possible by concentrating on these goals and addressing the complications that come with them as the system is created.

6.2 Escalating Paillier Cryptosystem to Starling Navigation

In this section, we explore the implementation process of incorporating the Paillier cryptosystem into the navigation protocols of multiple Starling-inspired families. For this implementation, we consider a fleet consisting of $N = 3$ families (A, B and C), each comprising three agents, thus totaling nine agents. The specific roles of each agent within the families are summarized and delineated in Table 6.1.

Agent 1A *Head Leader*	**Agent 1B** Follows *Head Leader*	**Agent 1C** Follows Agent 1B
Agent 2A Follows family leader (1A)	**Agent 2B** Follows family leader (1B)	**Agent 2C** Follows family leader (1C)
Agent 3A Follows Agent 2A	**Agent 3B** Follows Agent 2B	**Agent 3C** Follows Agent 2C

Table 6.1: Family Roles

Each family operates as a cohesive unit with a designated hierarchical structure to ensure coordinated movement and secure communication.

The process of adapting the already developed system of chapter 5, involved reforming some of the key steps, such that:

1. **Initial conditions of each agent**: The initial positions of the agents in each family were set to ensure that all agents had to deviate from their path, at some point due to an obstacle, thus being as follows:

Family A	**Family B**	**Family C**
Agent 1A: $[-2.0, 4.0]$	**Agent 1B**: $[-1.0, 1.0]$	**Agent 1C**: $[-3.0, 3.0]$
Agent 2A: $[-2.0, 2.0]$	**Agent 2B**: $[-1.0, 0.0]$	**Agent 2C**: $[-3.0, 2.0]$
Agent 3A: $[-2.0, 1.0]$	**Agent 3B**: $[-1.0, -1.0]$	**Agent 3C**: $[-3.0, 1.0]$

Table 6.2: Initial Conditions

2. **Bias terms**: The previous chapter elucidated the critical role of bias terms in ensuring data confidentiality and their intrinsic nature as characteristics of each agent. As the number of robots increases in this scenario, the number of bias terms correspondingly increases to maintain secure communication and coordination within and between families.

   Previously, bias terms such as b_{12}, b_{23}, and b_{32} were used to manage interactions between agents, a role that is kept in the current expanded framework expect that are now identified

by their respective "family names" (for example, b_{12} becomes b_{12A}). This designation helps in maintaining the unique communication pathways within each family.

Furthermore, the necessity for inter-family communication introduces additional bias terms. Specifically, the leaders of each family, designated as Agents 1A, 1B, and 1C, must communicate effectively with each other. To facilitate this, new bias terms such as b_{1A}, b_{1B}, and b_{1C} are introduced. These terms are essential for synchronizing the movements and ensuring the cohesive operation of the swarm at a higher hierarchical level.

3. **Key Generation and Encryption of Data**: This process is carried out in a manner consistent with the methodology described in the previous chapter, as follows:

```
from phe import paillier

################ FAMILY A ###############
public_key_1A, private_key_1A = paillier.generate_paillier_keypair()
public_key_2A, private_key_2A = paillier.generate_paillier_keypair()
public_key_3A, private_key_3A = paillier.generate_paillier_keypair()

enc_b12A = [public_key_2A.encrypt(x) for x in b12]
enc_b23A = [public_key_2A.encrypt(x) for x in b23]
enc_b32A = [public_key_3A.encrypt(x) for x in b32]

################ FAMILY B ###############
public_key_1B, private_key_1B = paillier.generate_paillier_keypair()
public_key_2B, private_key_2B = paillier.generate_paillier_keypair()
public_key_3B, private_key_3B = paillier.generate_paillier_keypair()

enc_b12B = [public_key_2B.encrypt(x) for x in b12]
enc_b23B = [public_key_2B.encrypt(x) for x in b23]
enc_b32B = [public_key_3B.encrypt(x) for x in b32]


enc_b1A = [public_key_1B.encrypt(x) for x in b1A]
enc_b1C = [public_key_1B.encrypt(x) for x in b1C]

################ FAMILY C ###############
public_key_1C, private_key_1C = paillier.generate_paillier_keypair()
public_key_2C, private_key_2C = paillier.generate_paillier_keypair()
public_key_3C, private_key_3C = paillier.generate_paillier_keypair()

enc_b12C = [public_key_2C.encrypt(x) for x in b12]
enc_b23C = [public_key_2C.encrypt(x) for x in b23]
enc_b32C = [public_key_3C.encrypt(x) for x in b32]

enc_b1B = [public_key_1C.encrypt(x) for x in b1B]
```

Listing 6.1: Public-Private Key Generation for each Agent and encryption of the bias terms

This process is implemented at the beginning of the simulation and needs to be run only once due to the inherent nature of the key pairs and bias terms. In contrast, each agent's positions are encrypted at each time sample.

4. **Computations**: The functions developed for the context of chapter 5 were seamlessly integrated and utilized as required for this implementation.

6.3 Results and Final Considerations

As in the previous chapters, the simulation was conducted in a 2D plane, with the initial coordinates for each agent as specified in the preceding section, with a sampling time of $h = 0.1s$ and $t\_end = 20$ simulation time.

Analyzing Figure 6.1, it can be inferred that the control algorithm, integrated with the Paillier cryptosystem, successfully guides all agents to accurately track their respective leader's trajectory[1]. This outcome reaffirms the algorithm's capability to maintain precise control despite the added complexity of encryption, demonstrating its effectiveness in securely managing agent trajectories within the simulation parameters.

Regarding the analysis of the time evolution of the three families, Figure 6.2, confirms that despite the increased number of agents in the environment, all were able to reach the same endpoint along the X-coordinate. This outcome demonstrates the system's capacity to maintain consistent inter-agent distances, as observed in the dashed lines of the figure. Such results highlight the robustness and effectiveness of the control algorithm in managing agent coordination and maintaining spatial consistency throughout the simulation.

However, much like the previous chapter, this one sees the primary goals achieved, particularly in addressing the challenge of system scalability. Despite these successes, the implementation remains impractical for real-world applications due to its increasingly problematic runtime. Notably, the runtime has deteriorated compared to the last chapter, largely because the number of agents has also increased. This escalation in agents exacerbates the already pressing issues, rendering the solution less feasible despite the advancements in scalability. Specifically, the time required for generating the key pairs and encrypting the bias terms is 55.5 seconds. Additionally, the overall time for executing the entire algorithm is approximately 99.16 minutes. These extensive time requirements highlight the significant efficiency challenges that need to be addressed for practical deployment.

[1] Video animation in https://drive.google.com/file/d/1-6ahj_2cD0TZEmURooUYz5o3X7XkmOwd/view?usp=sharing

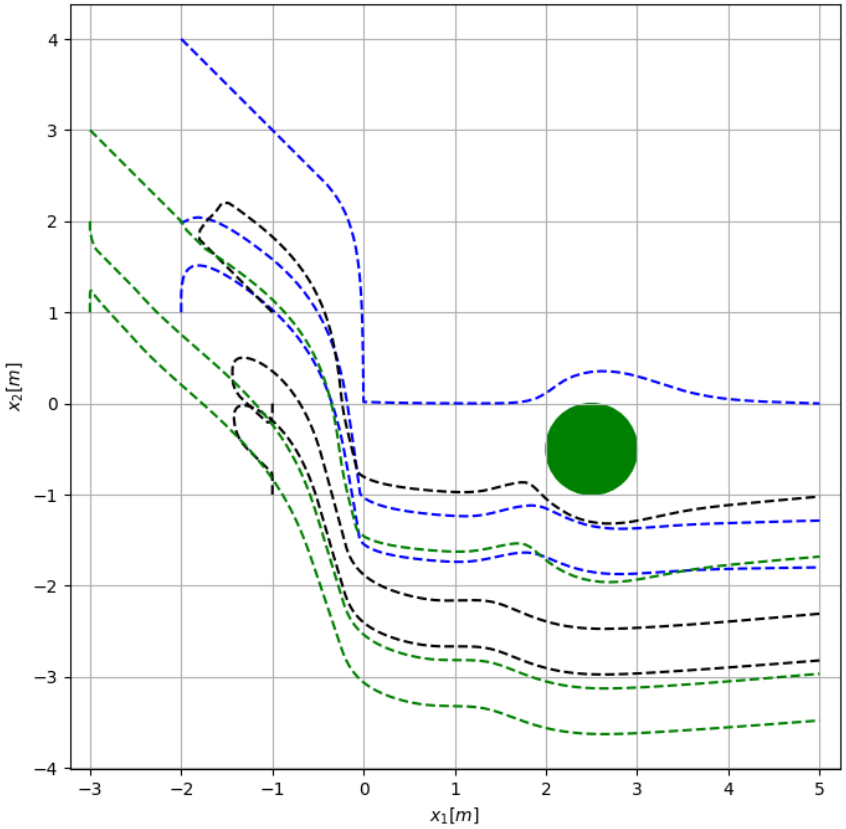


Figure 6.1: Trajectory of 3 encrypted starling families. Family A in blue, Family B in black and Family C in green.

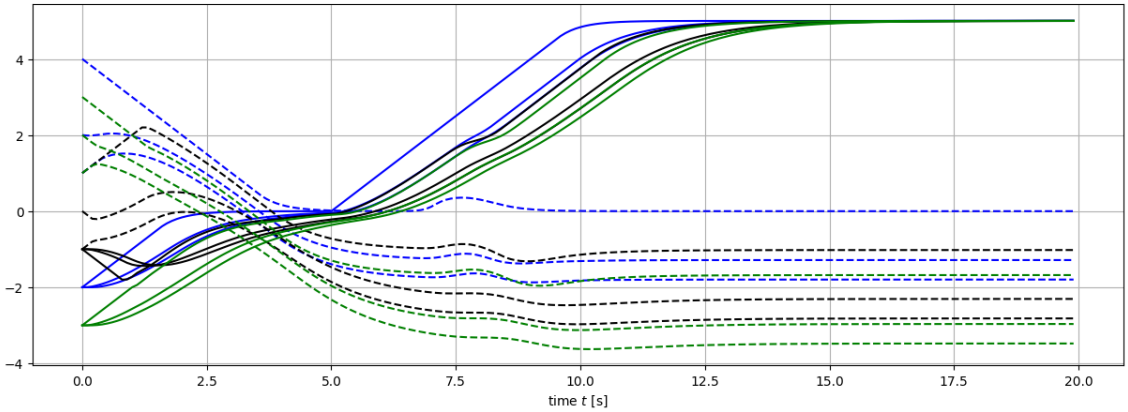


Figure 6.2: Time evolution of 3 encrypted starling families. Family A in blue, Family B in black and Family C in green.

Chapter 7

System Improvements

This chapter addresses the most significant challenge in implementing the Paillier Encryption Scheme within the domain of swarm robotics: its extensive long runtime, which renders it unsuitable for real-world applications. It is structured in two different sections, each presenting a distinct approach to mitigate this drawback and their respective outcomes.

The focus will be solely on the single-family scenario, ensuring a thorough examination of the proposed solutions.

7.1 Communication Delays

Let us consider a situation where each agent in the swarm announces its position after a predefined delay t. To illustrate this concept, let list a be defined as $a = [1,\ 2,\ 3,\ 4,\ 5]$ and list b as $b = a[i] + 1 = [2,\ 3,\ 4,\ 5,\ 6]$. In the context of a communication delay, list b should instead be constructed by taking the first element of a, adding one, and then "freezing" this value for t iterations before updating to the next corresponding value from a.

For instance, with $t = 2$, the list b is formed as follows: the first element of a is incremented by one, resulting in $b = [0] = 2$. This value remains constant for the next two iterations. The process then repeats with the next element of a, which is incremented by one and held for two iterations, and so on. Therefore, for $t = 2$, the resulting list b is $[2,\ 2,\ 4,\ 4, 6]$.

In the context of this dissertation, specifically, the single-family case, a is instead x_i, where i is $\{1,\ 2,\ 3\}$ - for Agents 1, 2 and 3 - and b is $u_i = -K_i \sum_{j \in N_i} \big((x_i - x_j) - b_{ij}\big)$. For illustrative purposes, the interaction between Agent 2 and Agent 1 when $t = 2$ is examined:

- In the first iteration the process is just as as depicted in Figure 3.6, and simplified in Figure 7.1.

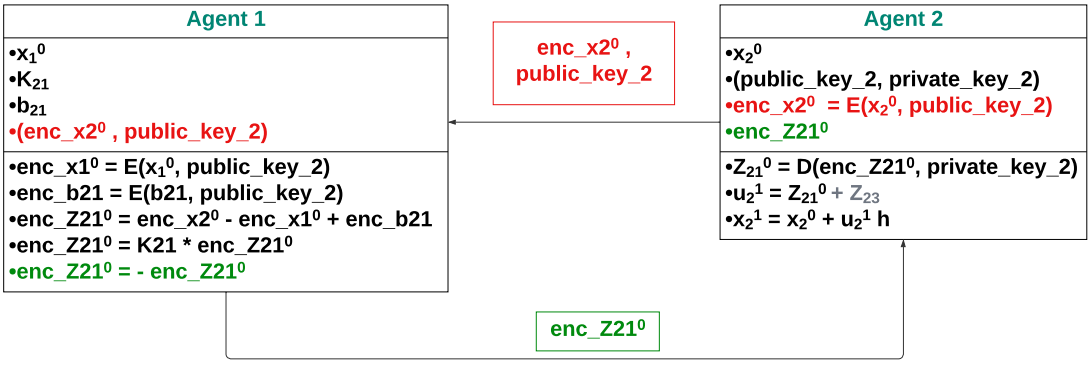


Figure 7.1: Conceptual diagram of communications between Agent 1 and Agent 2, in the first iteration.

- In the second iteration, however, rather than calculating Z_{21} using the current value of x_1 (i.e. x_1^1), Z_{21} is calculated using the previously "frozen" value of x_1 (i.e. x_1^0). This approach reduces the time each agent dedicates to encrypting each position, as fewer values need to be encrypted due to the reuse of the already encrypted value x_1^0. This process is illustrated in Figure 7.2.

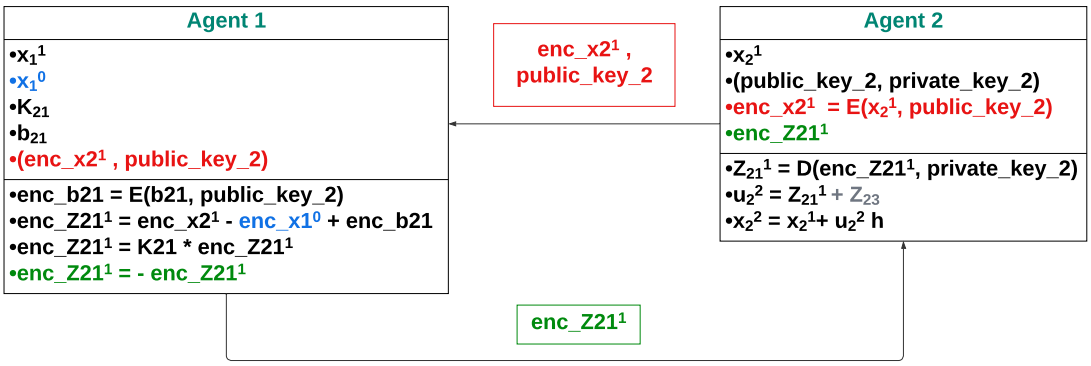


Figure 7.2: Conceptual diagram of communications between Agent 1 and Agent 2, in the second iteration.

7.1.1 Results

To evaluate the effectiveness of the proposed approach, the code was altered and the time required for encryption, computation, and decryption was recorded over 200 iterations, with delay intervals of $t = 2$, $t = 4$, and $t = 6$, using the same baseline values as described in Chapter 5.

For $t = 2$ iterations: Total time = 22.31 minutes

The recorded trajectories of all 3 Agents are depicted in Figure 7.3, where there are no noticeable differences when compared to the trajectories depicted in Figure 5.1.

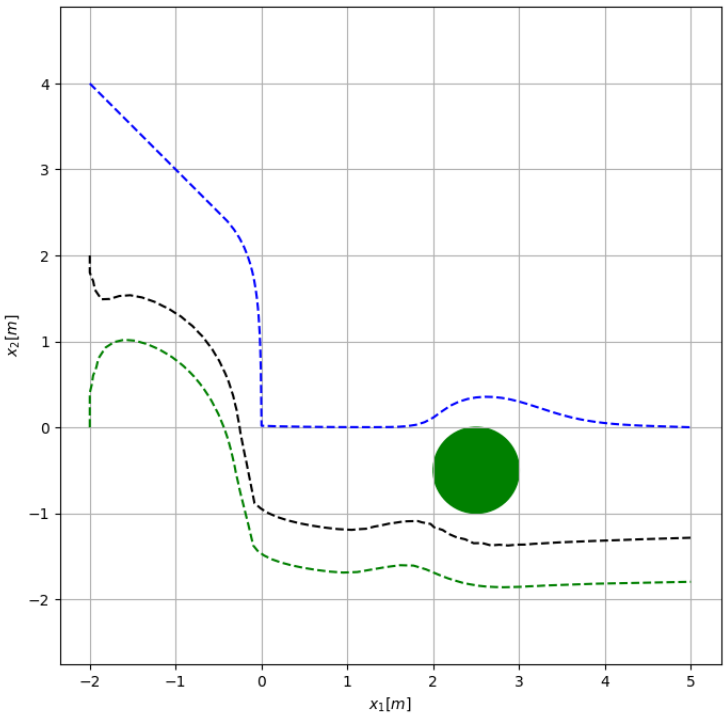


Figure 7.3: Trajectories of all Agents when there is a delay of positional encryption every 2 iterations.

When examining the plot depicting the time evolution of each agent in Figure 7.4, a slight "undulation" in x_2 and x_3 can be observed. However, the overall process remains similar to that recorded in Chapter 5, with all agents converging to the same X-value (represented by the continuous line) and the time required for all agents to reach this convergence remaining unchanged (approximately 14 seconds).

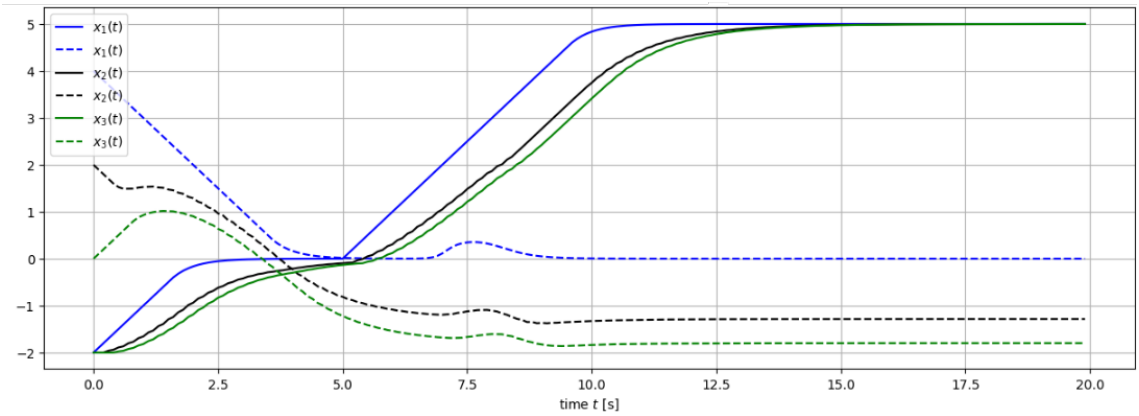


Figure 7.4: Time evolution of the state of each Agent when there is a delay of positional encryption every 2 iterations.

Figure 7.5 illustrates the time evolution of the distance between Agent 1 and Agent 2. A noticeable oscillation occurs around the 7.5 to 10-second mark, which arises because the x_1 values

used to compute the new u_2 are not as current as they were previously.

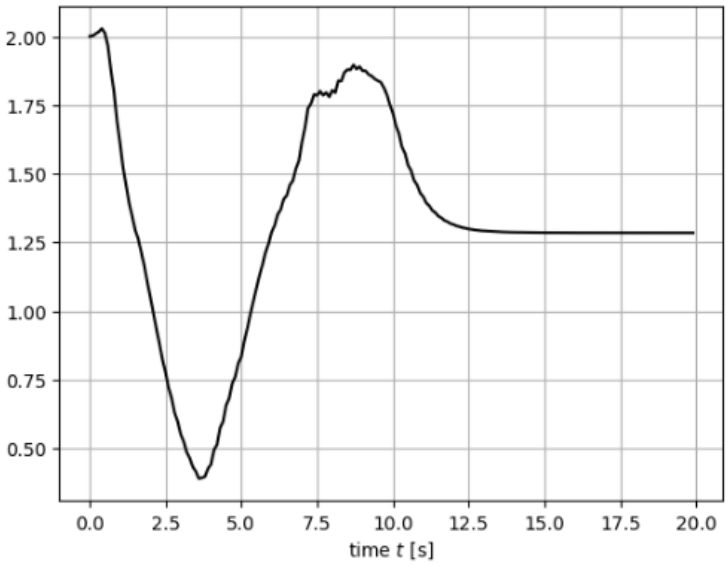


Figure 7.5: Time evolution of the distance between Agent 1 and Agent 2 when there is a delay of positional encryption every 2 iterations.

For $t = 4$ iterations: Total time = 20.64 minutes

Figure 7.6 presents the trajectories of all three agents, where the slight uncertainty in the trajectory of Agent 2 and Agent 3 becomes clear at the beginning of the course and close to the obstacle.

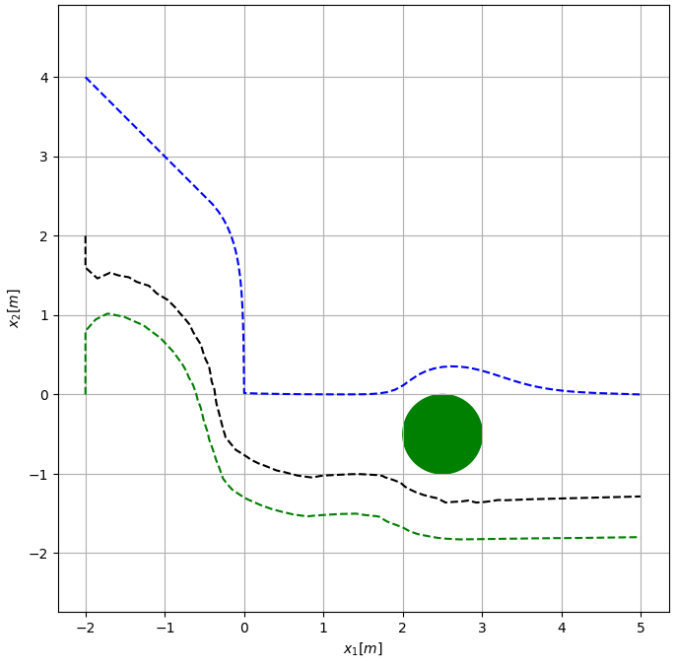


Figure 7.6: Trajectories of all Agents when there is a delay of positional encryption every 4 iterations.

Depicted in Figure 7.7, the oscillations observed when $t = 2$ become even more pronounced as the delay t increases. This is evident in both the dashed line, representing the Y-values of each agent, and the solid line, representing the X-values.

Additionally, the convergence time experienced a noticeable setback, now exceeding 15 seconds. This decline in performance was anticipated, as the increasing delay requires more time for agents to obtain accurately encrypted information from each other. The delay inherently introduces a lag in the update process, thereby affecting the overall system's responsiveness and efficiency. However, despite this increased convergence time, all agents successfully achieved their primary objective: reaching their designated end destinations while avoiding collisions with each other and obstacles.

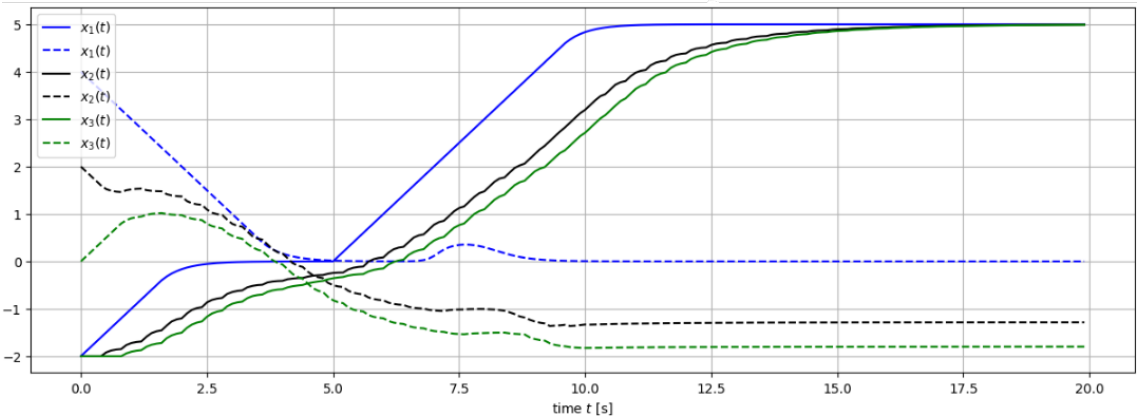


Figure 7.7: Time evolution of the state of each Agent when there is a delay of positional encryption every 4 iterations.

Furthermore, when focusing on the distance between Agent 2 and Agent 1, in Figure 7.8, it is evident that this distance increases between the period of 7.5 seconds and 10 seconds, accompanied by minor oscillations. Despite these fluctuations, the agents ultimately achieve the desired separation distance by the end of the period.

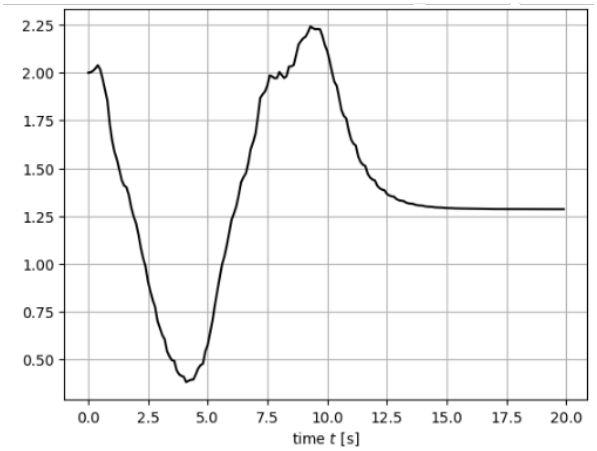


Figure 7.8: Time evolution of the distance between Agent 1 and Agent 2 when there is a delay of positional encryption every 4 iterations.

For $t = 6$ iterations: Total time = 19.46 minutes

As expected, with the increase of t, the trajectories of Agent 2 and Agent 3, while still achieving the primary goal, become increasingly irregular, as seen in Figure 7.9.

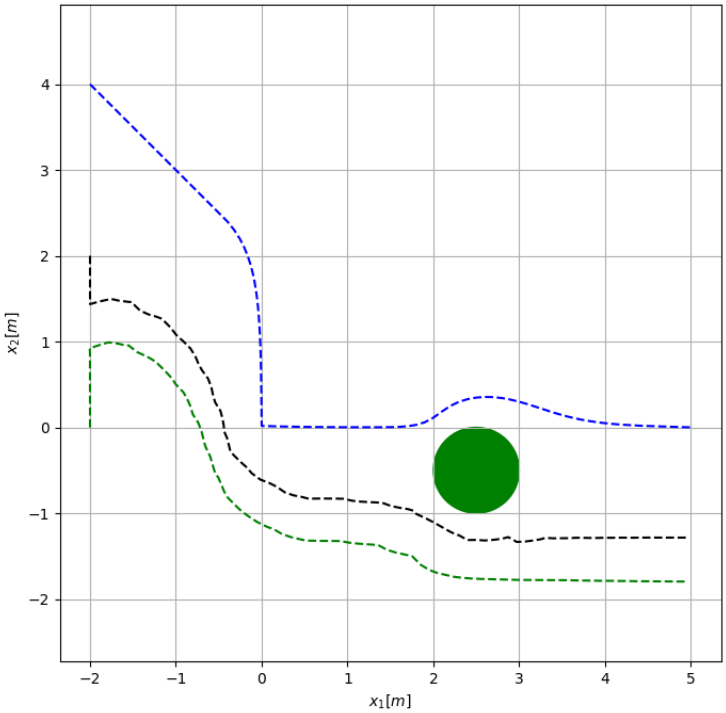


Figure 7.9: Trajectories of all Agents when there is a delay of positional encryption every 6 iterations.

The convergence time experiences yet another noticeable setback, now requiring approximately 19 seconds for all agents to reach their end destination (as depicted in Figure 7.10).

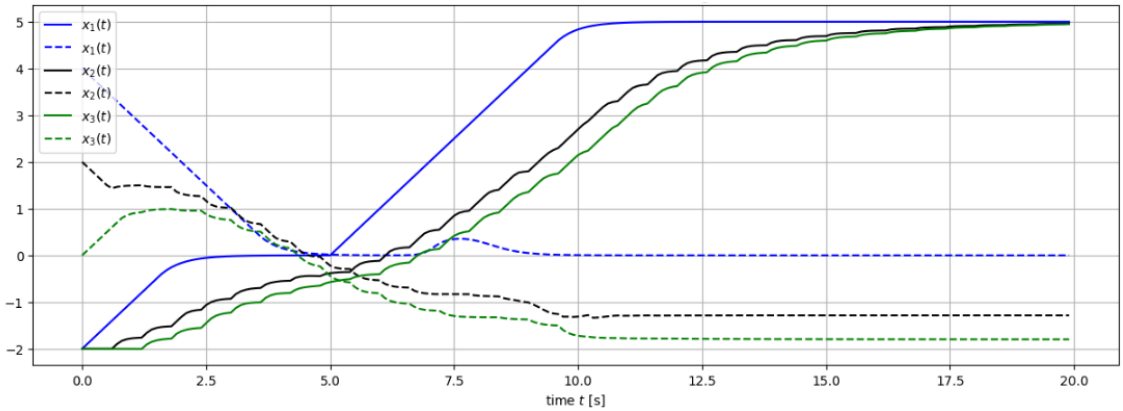


Figure 7.10: Time evolution of the state of each Agent when there is a delay of positional encryption every 6 iterations.

In an analogous manner, the plot's curvature gets worse as Agents 1 and 2 get farther apart (depicted in Figure 7.11).

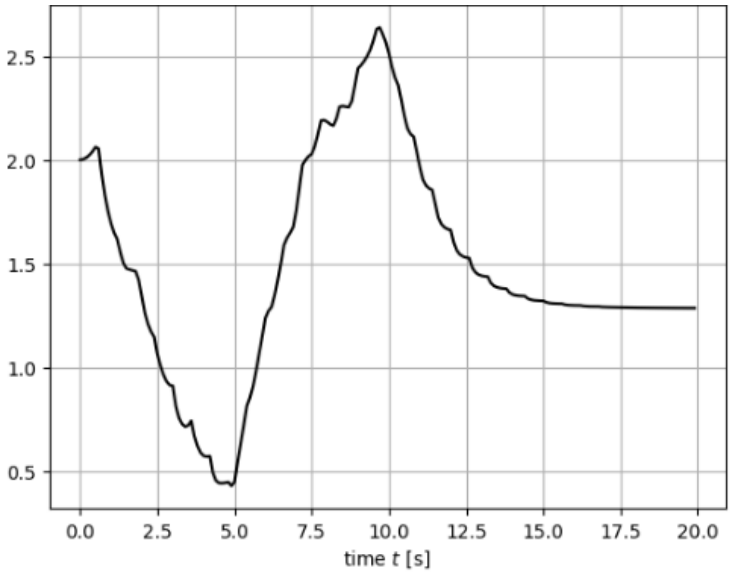


Figure 7.11: Time evolution of the distance between Agent 1 and Agent 2 when there is a delay of positional encryption every 6 iterations.

7.2 Intermittent Communications

This section builds upon the approach discussed previously, which aimed to reduce the total runtime for encryption, computation, and decryption by decreasing the frequency of positional updates. In the earlier approach, agents communicated their positions less frequently, thereby reducing the computational load associated with constant encryption.

This section takes this idea a step further by proposing a more radical reduction in communication frequency. Specifically, the impact of completely suspending communication between agents for a predefined number of iterations, denoted as t_i, is explored. This approach is expected to yield a significantly reduced computational burden and improve runtime efficiency by eliminating the need for inter-agent communication during these intervals.

7.2.1 Results

For $t_2 = t_3 = 2$ iterations : Total time = 13.03 minutes

In this scenario, Agent 2 communicates with Agent 1 and Agent 3 for only 2 iterations at a time, providing them with the encrypted result of its position along with the respective public key ($t_2 = 2$). The same communication pattern is applied to the link between Agent 3 and Agent 2 when Agent 3 needs to obtain the result of u_3 ($t_3 = 2$).

Figure 7.12 illustrates the trajectories of the three agents under the same conditions as previously described. Compared to Figure 5.1, in chapter 5, the performance remains consistent, with no visible alterations. All agents successfully reach their designated endpoints, following the same paths as in the encrypted non-optimal system detailed earlier, in less time.

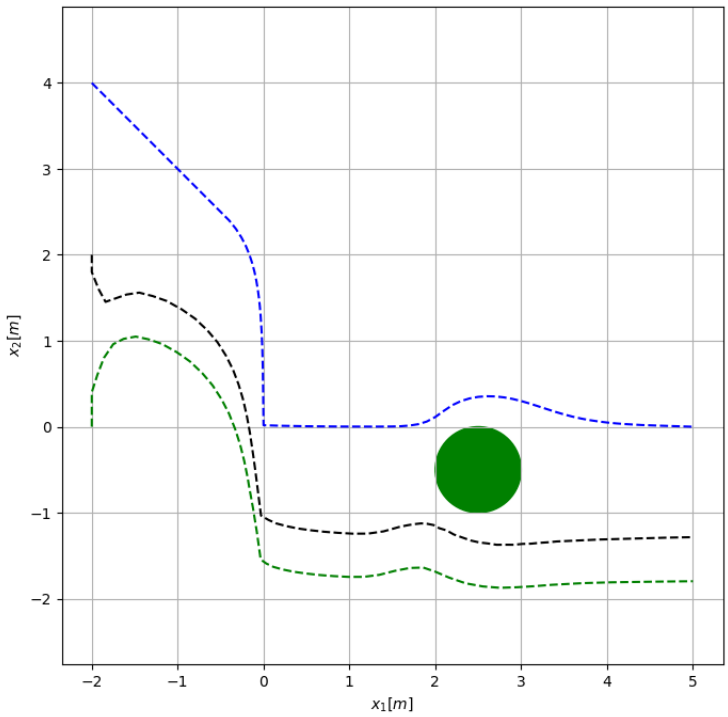


Figure 7.12: Trajectories of all Agents when there is an interruption of communications every 2 iterations for both Agent 2 and Agent 3.

A similar conclusion can be conducted for Figures 7.13 and 7.14, as these figures show no discernible differences in performance when compared to the non-optimal encrypted version, with the exception of a significant reduction in runtime.

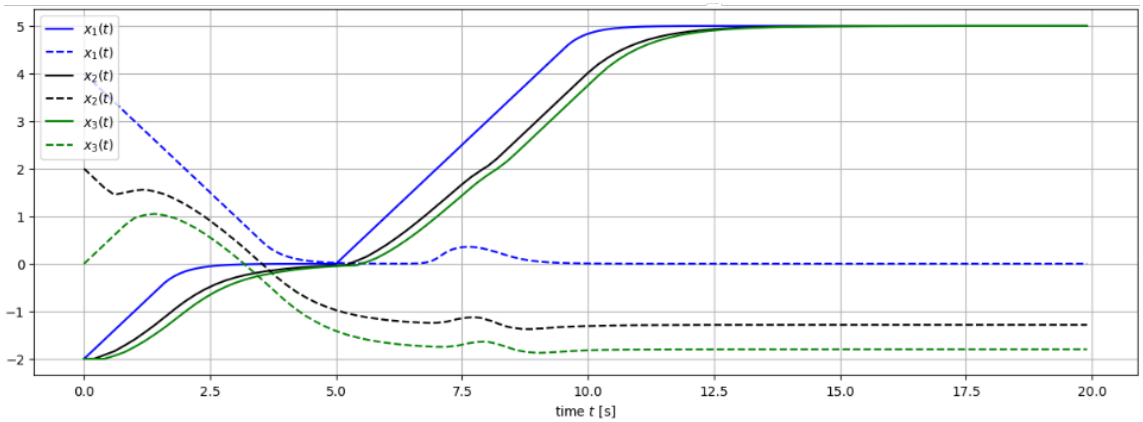


Figure 7.13: Time evolution of the state of each Agent when there is an interruption of communications every 2 iterations for both Agent 2 and Agent 3.

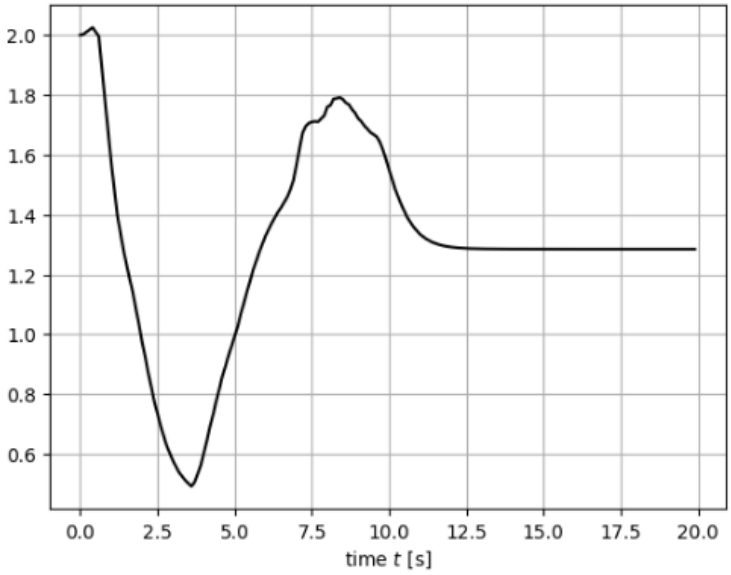


Figure 7.14: Time evolution of the distance between Agent 1 and Agent 2 when there is an interruption of communications every 2 iterations for both Agent 2 and Agent 3.

This consistency demonstrates that reducing communication frequency does not negatively impact the coordination and goal attainment of the swarm. Despite fewer updates, the agents navigate effectively and avoid collisions, indicating that the system's robustness is preserved.

For $t_2 = 3$ iterations and $t_3 = 5$ iterations: Total time = 8.18 minutes

To determine the maximum value of t at which the total runtime remains lower, and the performance is not significantly affected, a scenario was simulated where Agent 2 communicated with Agent 1 and Agent 3 for 3 iterations at a time to update the value of u_2 ($t_2 = 3$). Similarly, Agent 3 communicated with Agent 2 for 5 iterations at a time to update u_3 ($t_3 = 5$).

The outcomes of this implementation are presented in Figures 7.15, 7.16 and 7.17. The most notable performance variation is seen in Figure 7.15, especially as the agents navigate the obstacle between coordinates $[1,-1]$ and $[2,-1]$. Despite this visible difference in trajectory, the convergence time for all agents to reach their final positions remains consistent, as evidenced by Figure 7.16. Furthermore, the distance between Agent 1 and Agent 2 remains stable, as illustrated in Figure 7.17.

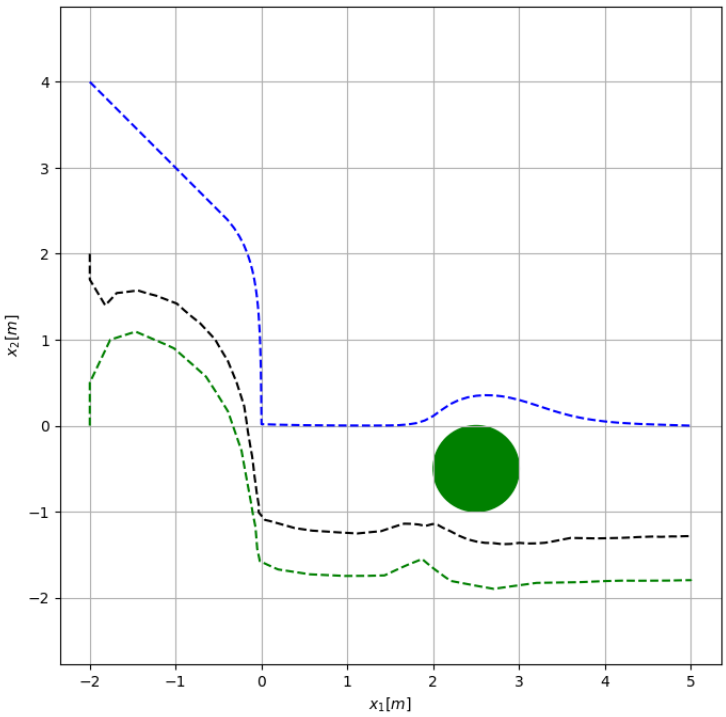


Figure 7.15: Trajectories of all Agents when there is an interruption of communications every 3 iterations between Agent 1 and Agent 2 and an interruption between Agent 2 and Agent 3 every 5 iterations.

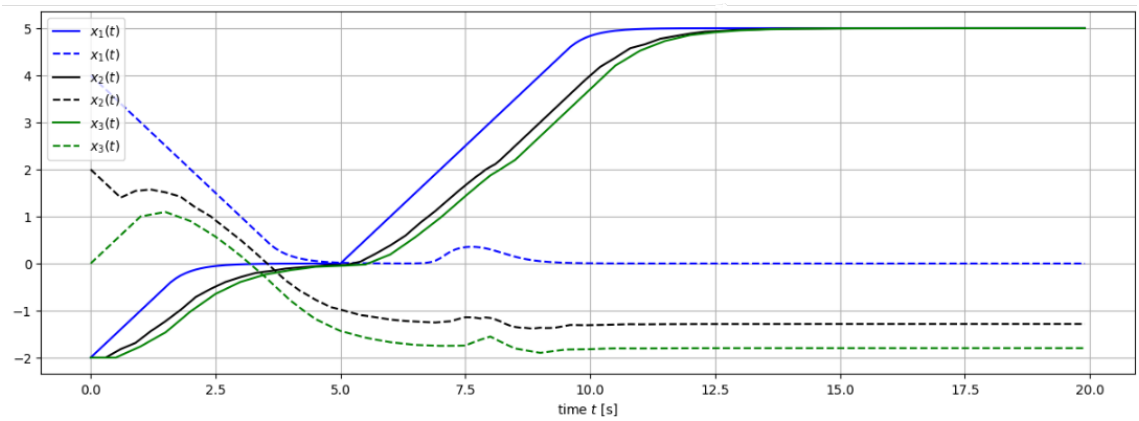


Figure 7.16: Time evolution of the state of each Agent when there is an interruption of communications every 3 iterations between Agent 1 and Agent 2 and an interruption between Agent 2 and Agent 3 every 5 iterations.

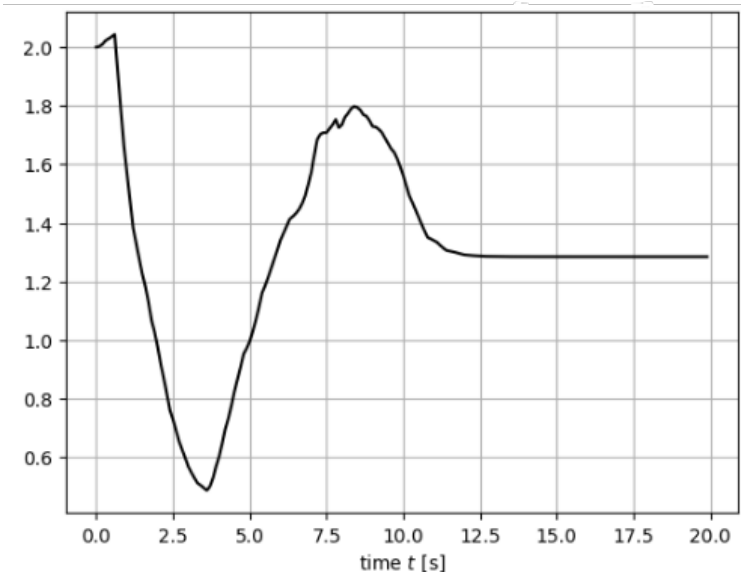


Figure 7.17: Time evolution of the distance between Agent 1 and Agent 2 when there is an interruption of communications every 3 iterations between Agent 1 and Agent 2 and an interruption between Agent 2 and Agent 3 every 5 iterations.

7.3 Discussion

In this chapter, we explored two distinct approaches to mitigate the extensive runtime associated with the implementation of the Paillier Encryption Scheme in swarm robotics. Each approach presented its unique set of benefits and trade-offs, shedding light on the ongoing challenge of balancing performance, efficiency, and security in real-world applications.

In the first implementation, we introduced a communication delay t to reduce the computational burden on the agents. As t increased, the overall runtime of the program decreased significantly. This reduction in runtime is attributed to the less frequent need for encryption and decryption operations, which are computationally intensive. However, this improvement in computational efficiency came at the cost of performance quality.

Specifically, as the delay t increased, the trajectories of the agents became more erratic and the system's convergence time increased. The agents still achieved their primary goal of reaching their destinations and avoiding collisions, but the path taken was less optimal.

The second implementation introduced a strategy where communications between agents were paused for a certain number of iterations. Specifically, Agent 2 only communicated with Agent 1 and Agent 3 every 3 iterations, and Agent 3 only communicated with Agent 2 every 5 iterations. This approach significantly reduced the frequency of encrypted communications, thereby lowering the computational load.

As a result, the system maintained the efficiency of the non-encrypted version, with the total runtime reduced to just 8.18 minutes. This is a substantial improvement compared to the initial implementation discussed in Chapter 5, where the total runtime was over 25 minutes. The agents not

only reached their destinations successfully but also had more predictable and stable trajectories than the first improvement implementation of this chapter.

Nevertheless, both implementations highlight a fundamental engineering dilemma: the trade-off between performance, time efficiency, and data security. Neither solution can achieve the near-instantaneous response time of the non-encrypted system detailed in Chapter 3 Thus a critical decision must be made: in scenarios where time efficiency and rapid response are paramount, it might be more practical to forego encryption. However, in situations where these conditions can be relaxed, the second approach presented in this chapter offers a viable alternative, balancing the need for security with efficiency

Ultimately, the decision of which approach to adopt depends on the specific priorities and constraints of the given situation.

Chapter 8

Conclusions and Future Work

This dissertation set out to explore and extend the application of homomorphic encryption, specifically Paillier's encryption scheme, to the domain of swarm robotics, drawing inspiration from the complex, yet elegant, flocking behavior of starlings.

The research began with establishing a theoretical framework by synthesizing cryptographic principles, specifically focusing on Paillier's encryption scheme due to its additive homomorphic properties. This framework formed the basis for integrating homomorphic encryption into the communication protocols of swarm agents, ensuring secure and private data exchange.

A system architecture was developed to support this secure communication framework. Motion control algorithms were then designed, enabling swarm agents to navigate toward target positions while avoiding obstacles. These algorithms were validated in both encrypted and non-encrypted scenarios, providing a baseline performance metric for comparison.

The introduction of Paillier's encryption into the control laws was a critical phase, in assessing the performance of the encrypted system. The results showed that while the accuracy of the agents' trajectories remained consistent, the efficiency—measured in terms of runtime — was impacted. The encrypted system took longer to achieve its objectives compared to the non-encrypted baseline, highlighting the computational challenges associated with homomorphic encryption.

The research was further extended by adapting the control and encryption mechanisms to accommodate multiple families of robots. This extension demonstrated the scalability of the control algorithm, showing that multiple robots could achieve coordinated movement towards a common goal, albeit with some efficiency challenges.

8.1 Future Work

The findings of this dissertation suggest several avenues for future research that could enhance the efficiency and practicality of secure swarm robotics:

1. **Exploring Alternative Encryption Schemes:** Future research could investigate the use of different homomorphic encryption schemes to see if they offer improved efficiency over

Paillier's scheme. Schemes that are more computationally efficient could reduce the overhead and latency observed in this study, thereby improving the overall performance of the swarm system.

2. **Implementing Real-World Scenarios:** Another important direction is to test the proposed system in real-world scenarios. Implementing a practical setup where multiple robots communicate with each other to achieve a specific goal would provide valuable insights into the challenges and feasibility of deploying such systems outside of simulated environments. This could involve navigating through dynamic, obstacle-filled spaces or performing coordinated tasks, such as in a platooning system.

3. **Improving Paillier Encryption Efficiency:** There is potential to enhance the efficiency of Paillier's encryption scheme itself. Research could focus on algorithmic optimizations or hardware acceleration techniques that reduce computational load and speed up encryption and decryption processes. Such improvements would directly address the performance bottlenecks identified in this dissertation.

Pursuing these research directions could significantly advance the field of secure swarm robotics, making it more viable for real-world applications while maintaining robust privacy and security standards.

In summary, this dissertation has made significant strides in bridging the gap between cryptographic security and swarm robotics, demonstrating that homomorphic encryption can be effectively applied to ensure secure coordination among robotic agents. While challenges remain, particularly in terms of efficiency, the foundational work laid out here opens numerous possibilities for future research and practical applications. By continuing to explore and refine these techniques, we can move closer to realizing secure, efficient, and intelligent swarm robotic systems capable of tackling complex tasks in a variety of environments, much like the harmonious and efficient movements of a starling flock.

References

[1] Mohamed Abdelkader, Samet Güler, Hassan Jaleel, and Jeff S. Shamma. Aerial swarms: Recent applications and challenges. *Current Robotics Reports*, 2:309–320, 9 2021.

[2] Mohamed Abdelkader, Mohammad Shaqura, Mehdi Ghommem, Nathan Collier, Victor Calo, and Christian Claudel. Optimal multi-agent path planning for fast inverse modeling in uav-based flood sensing applications. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 64–71, May 2014.

[3] Leah Abraham, Sibi Biju, Felix Biju, Jithin Jose, Rakhi Kalantri, and Shahgufta Rajguru. Swarm robotics in disaster management. In *2019 International Conference on Innovative Sustainable Computational Technologies (CISCT)*, pages 1–5, 2019.

[4] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), jul 2018.

[5] Dario Albani, Joris IJsselmuiden, Ramon Haken, and Vito Trianni. Monitoring and mapping with robot swarms for agricultural applications. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, Aug 2017.

[6] Antonio Luca Alfeo, Eduardo Castelló Ferrer, Yago Lizarribar Carrillo, Arnaud Grignard, Luis Alonso Pastor, Dylan T. Sleeper, Mario G.C.A. Cimino, Bruno Lepri, Gigliola Vaglini, Kent Larson, Marco Dorigo, and Alex Pentland. Urban swarms: A new approach for autonomous waste management. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4233–4240, May 2019.

[7] Liqun Chen and Siaw-Lynn Ng. Securing emergent behaviour in swarm robotics. *Journal of Information Security and Applications*, 64:103047, 2022.

[8] Marco Dorigo, Guy Theraulaz, and Vito Trianni. Swarm robotics: Past, present, and future [point of view]. *Proceedings of the IEEE*, 109(7):1152–1165, July 2021.

[9] Eduardo Castelló Ferrer, Thomas Hardjono, Alex Pentland, and Marco Dorigo. Secure and secret cooperation in robot swarms. *Sci. Robot*, 6:1–9, 2021.

[10] Eduardo Castelló Ferrer, Thomas Hardjono, Alex Pentland, and Marco Dorigo. Secure and secret cooperation in robot swarms. *Science Robotics*, 6(56):eabf1538, 2021.

[11] Fiona Higgins, Allan Tomlinson, and Keith M. Martin. Survey on security challenges for swarm robotics. In *2009 Fifth International Conference on Autonomic and Autonomous Systems*, pages 307–312, April 2009.

[12] Yixin Huang, Shufan Wu, Zhongcheng Mu, Xiangyu Long, Sunhao Chu, and Guohong Zhao. A multi-agent reinforcement learning method for swarm robots in space collaborative exploration. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 139–144, 2020.

[13] Shiho Kim and Rakesh Shrestha. *Automotive Cyber Security Introduction, Challenges, and Standardization*. Springer Nature Singapore Pte Ltd, 152 Beach Road, 21-01/04 Gateway East, Singapore 189721, Singapore, 2021.

[14] Nitin Kohli. A quick example of homomorphic encryption using the paillier cryptosystem: Supplemental note, 2024.

[15] Li Li, Alfredo Bayuelo, Leonardo Bobadilla, Tauhidul Alam, and Dylan A. Shell. Coordinated multi-robot planning while preserving individual privacy. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2188–2194, May 2019.

[16] Alfred J. Menezes, Paul C van Oorschot, and Scott A. Vanstone. Overview of cryptography contents in brief.

[17] Russell Paulet, Elisa Bertino, and Xunn Yi. *Homomorphic Encryption and Applications*. Springer, Cham, 2014.

[18] Jaydip Sen. Homomorphic encryption-theory and application. *Theory and practice of cryptography and network security protocols and technologies*, 31, 2013.

[19] Muhammad Muzamal Shahzad, Zubair Saeed, Asima Akhtar, Hammad Munawar, Muhammad Haroon Yousaf, Naveed Khan Baloach, and Fawad Hussain. A review of swarm robotics in a nutshell. *Drones*, 7(4), 2023.

[20] Petter Solnør, Slobodan Petrovic, and Thor I. Fossen. Towards oblivious guidance systems for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 72, 2023.

[21] Petter Solnør, Slobodan Petrovic, and Thor I. Fossen. Towards oblivious guidance systems for autonomous vehicles. *IEEE Transactions on Vehicular Technology*, 72(6):7067–7081, June 2023.

[22] Tanyaporn Sridokmai and Somchai Prakancharoen. The homomorphic other property of paillier cryptosystem. In *2015 International Conference on Science and Technology (TICST)*, pages 356–359, Nov 2015.

[23] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Managing byzantine robots via blockchain technology in a swarm robotics collective decision making scenario, 2018.

[24] Volker Strobel, Eduardo Castelló Ferrer, and Marco Dorigo. Blockchain technology secures robot swarms: A comparison of consensus protocols and their resilience to byzantine robots. *Frontiers in Robotics and AI*, 7, 5 2020.

[25] Guang Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, Bradley J. Nelson, Brian Scassellati, Mariarosaria Taddeo, Russell Taylor, Manuela Veloso, Zhong Lin Wang, and Robert Wood. The grand challenges of science robotics. *Science Robotics*, 3, 1 2018.