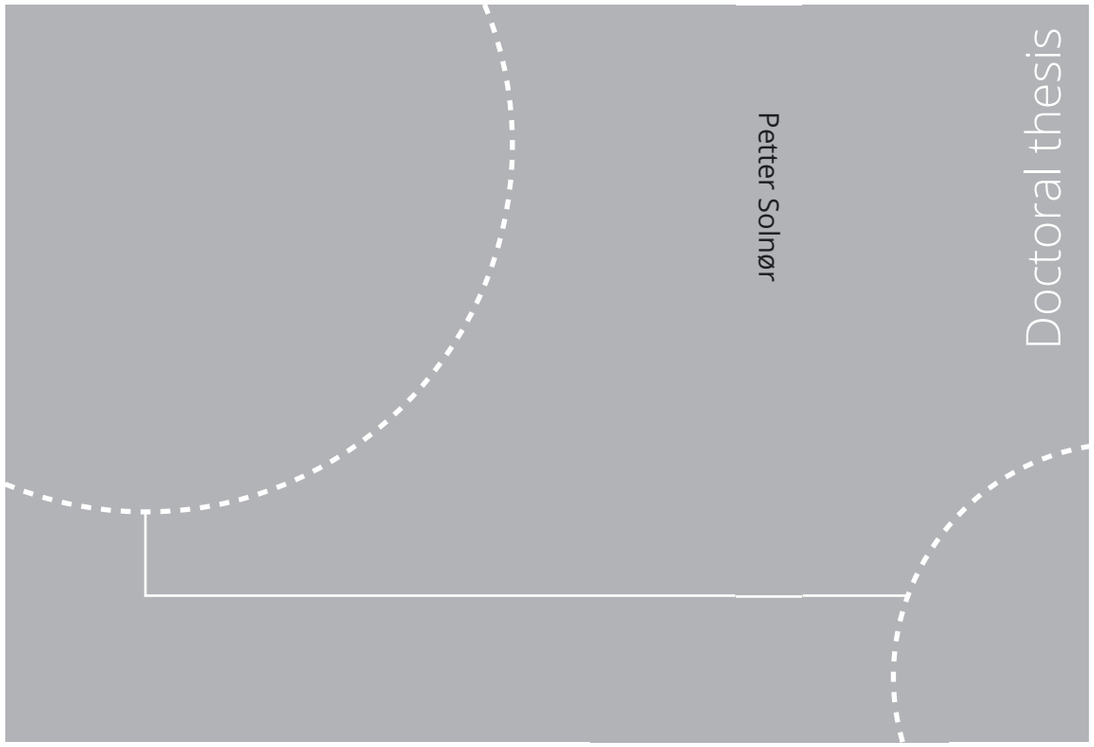
ISBN 978-82-326-7106-9 (printed ver.)
ISBN 978-82-326-7105-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (electronic ver.)



Doctoral theses at NTNU, 2023:202

Petter Solnør

Applications of Cryptographic Methods in Feedback Control

NTNU
Norwegian University of
Science and Technology
Thesis for the degree of
Philosophiae Doctor
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Doctoral theses at NTNU, 2023:202

NTNU

NTNU
Norwegian University of
Science and Technology

Petter Solnør

Applications of Cryptographic Methods in Feedback Control

Thesis for the degree of Philosophiae Doctor

Trondheim, June 2023

Norwegian University of Science and Technology
Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-7106-9 (printed ver.)
ISBN 978-82-326-7105-2 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (electronic ver.)

ITK-report: 2020-13-W

Doctoral theses at NTNU, 2023:202

Printed by Skipnes Kommunikasjon AS

To my family

Summary

Advances in information and communication technologies drive rapid developments in and increased adoption of unmanned and autonomous vehicles, desirable in numerous applications, such as public transportation, shipping, environmental mapping, and remote surveillance systems. Onboard these vehicles are increasingly modular guidance, navigation, and control systems connected across general-purpose networks, often called networked control systems, enabling more flexible hardware- and software architectures, reducing maintenance costs, and ensuring ease of installation. Moreover, increased connectivity and cloud computing technology permit outsourcing computational tasks to cloud-computing services. However, this increased connectivity also introduces new challenges concerning cybersecurity. Insecure communication channels make systems vulnerable to cyberattacks such as data injection, spoofing, and replay attacks. Similarly, outsourcing computations to third-party cloud servers may cause leaks of confidential system information.

Throughout this thesis, we are concerned with securing communication links onboard these vehicles and designing privacy-preserving systems with which we can outsource computations without exposing private information to inadvertent leaks. To this end, the thesis is composed of two parts. In the first part, we are concerned with secure signal transmission between distributed components in networked control systems and conventional cybersecurity. The second part of the thesis presents methods to design privacy-preserving control and guidance systems and information fusion schemes.

We start the first part of the thesis by considering using stream ciphers and authenticated encryption to obtain secure and computationally efficient data transmission in distributed guidance, navigation, and control systems connected over multi-purpose networks. We are motivated by the observation that previous studies have suggested using various ad-hoc constructions to achieve authenticated encryption for secure signal transmission between the components of networked control systems. However, these constructions consist of compositions of block ciphers and legacy algorithms with questionable efficiency and security. To this end, we show how we can use modern stream ciphers and cryptographically strong authenticated encryption to achieve secure and efficient data transfer between computing devices in distributed guidance, navigation, and control architectures. Through experimental validation of a cryptographic pipeline in the Robot Operating System, we show that modern stream ciphers perform very well on sensor data such as images and point clouds, which require significant throughput. Hence, using these algorithms should enhance security without adversely affecting the overall performance of the

closed-loop system. We also consider the use of 'compress-then-encrypt' schemes and show that compression should only be used if the bandwidth is constrained.

We then consider potential cyber-physical attacks against vehicles with distributed guidance, navigation, and control architectures. To this end, we use an unmanned surface vehicle where the navigation and the guidance & control systems run on different computing devices. By spoofing the address resolution protocol, we redirect the signal transmission through a device under our control, where we can change the position and heading estimates from the navigation device. These changes then result in predictable changes in the vehicle's path. Since the only integrity check used by the communication protocol consists of un-keyed cyclic redundancy checks, we can recompute the cyclic redundancy check of the manipulated messages such that the attack goes undetected by the guidance & control system. We show how to prevent such attacks by imposing authenticated encryption on the communication links. We also demonstrate that auxiliary information, such as timestamps, is essential to detect replay attacks.

In the second part of the thesis, we consider the design, implementation, and experimental validation of privacy-preserving systems which we can host on cloud infrastructure without leaking information. To this end, we use a cryptographic concept called homomorphic encryption to design *encrypted* systems that perform computations directly on encrypted data. We start the second part of the thesis by presenting and implementing an encrypted control system for the surge speed and yaw of an unmanned surface vehicle that computes encrypted thrust allocations over encrypted control gains and state information. We achieve this using a cryptosystem called labeled homomorphic encryption, which allows both homomorphic additions and homomorphic multiplications at the cost of revealing the function we are evaluating. We then validate the effectiveness of the encrypted control system through field experiments on an unmanned surface vehicle in the Trondheim Fjord, where it is exposed to considerable environmental disturbances.

We proceed by conceptualizing, designing, and implementing encrypted guidance systems. The motivation is that guidance systems are outer-loop systems that are iterated less frequently than inner-loop control systems that compute thrust allocations. Therefore, we argue that it is more intuitive to outsource guidance systems than encrypted control systems. To this end, we show that by revealing the bearing between individual waypoints and linearizing the line-of-sight and integral line-of-sight guidance laws, we can use an additively homomorphic cryptosystem to design encrypted guidance systems with and without integral action. These guidance systems operate with plaintext gains but compute encrypted course and heading commands using encrypted position measurements and waypoints. Hence, the host of the guidance system cannot derive the vehicle's position and planned path. We show that these guidance laws are locally exponentially stable and argue that, in practice, local stability is often sufficient for guidance laws since the vehicles tend to stay close to their desired paths. Finally, we demonstrate the effectiveness of an encrypted system with integral action through field experiments on the Trondheim Fjord using an unmanned surface vehicle.

Finally, we consider the fusion of encrypted unbiased Gaussian estimates using a concept called encrypted fast covariance intersection. Fusing estimates directly in encrypted form can be desirable for numerous reasons; for example, we present a

collaborative air defense surveillance system as a potential use case. Countries may want to collaborate to obtain a more accurate, fused position estimate of a target. However, sharing radar data may not be possible since the accuracy of individual radar stations may be considered classified. To this end, encrypted fast covariance intersection allows sharing and fusing estimates without revealing how accurate each estimate is and, thus, how well each sensor system performs. A variant of encrypted fast covariance intersection already exists. However, we show how we can use stream ciphers and privacy-preserving aggregation to design accelerated and decentralized variants of the scheme, respectively. Using simulations, we demonstrate that the accelerated variant is approximately five to six orders of magnitude faster than the existing algorithm. In addition, for a 128-bit level of security, we show that the accelerated variant reduces the amount of data transmitted by approximately 99%. The decentralized scheme is slower than the original scheme but eliminates the need for a fusion center and the assumption that all recipients must be honest. We also demonstrate that the performance of the two variants is identical to the existing scheme in terms of the accuracy of the fused output.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of philosophiae doctor (Ph.D.) at the Norwegian University of Science and Technology (NTNU), Trondheim. The research has been conducted at the Department of Engineering Cybernetics (ITK).

Professor Thor I. Fossen at the Department of Engineering Cybernetics has been the main supervisor of this work, while Professor Slobodan Petrović at the Department of Information Security and Communication Technology has co-supervised the work. The research has been funded by the Research Council of Norway through the Autonomous Marine Operations and Systems (AMOS) centre of excellence funding scheme, project number 223254.

Acknowledgments

Throughout my time at the department, I have met numerous people that have assisted me in one way or another. First and foremost, I would like to thank my main supervisor, Professor Thor I. Fossen, for initiating this project. An 'integrated Ph.D.' seemed like a considerable commitment at the time, but after our initial meeting in your office, I had no doubt this project would be a lot of fun. Throughout, you have contributed with numerous visionary ideas and infectious enthusiasm. But most importantly, you have given me near-total freedom to pursue whatever ideas we came across, making these years a rewarding and enjoyable journey. Second, I would like to thank my co-supervisor, Professor Slobodan Petrović, for welcoming me to Gjøvik and introducing me to the field of cryptology. Your insight, historical anecdotes, and patience with a novice made it a very enlightening experience. Moreover, without your detailed feedback on manuscripts and impeccable grammar, the peer review process would undoubtedly have been a rougher experience.

Next, I would like to thank my closest colleague over the years, Øystein Volden. I remember our first discussion concerning this project in the spring of 2019 when we both agreed that this seemed like a great opportunity. Since then, we have collaborated on several papers and spent countless hours conceptualizing and discussing ideas, debugging code, and mounting hardware. Always positive and focused on the task at hand, you ensured that we both stayed productive and never strayed too far off-topic. We also had a great stay in Lisbon that, quite frankly, you initiated. It was a blast, and I could not have asked for a better colleague and office mate.

Experimental validation has been a cornerstone in several of my publications. These field trips would not have been possible without the assistance of several helpful people in the department. First, I would like to thank Kristoffer Gryte from the UAVlab. Without your assistance during those early months of 2021, sharing your expertise in the LSTS toolchain, embedded systems, and field experiments, we would not have been able to set up the experimental platform as fast as we did. Another key person here is Nikolai Lauvås, who did much preliminary software development and implemented many features for his own experimental platforms that we could reuse. Without you, much time would be lost, and for that, I am grateful. I would also like to thank Terje Haugen and Glenn Angell at the mechanical workshop. You always provide quick assistance and fix things whenever they inevitably break, regardless of your schedule, something I know researchers at other departments envy.

During the fall of 2022, I visited the Institute for Systems and Robotics at the Instituto Superior Técnico. I am very grateful for the kind reception by Professor António Pascoal and Professor David Cabecinhas and all the students and researchers I met and got to know in the Dynamical Systems and Ocean Robotics laboratory. It was an incredible experience.

I believe a stimulating work environment is essential to conducting the creative work that is research. To this end, I would also like to thank everyone who contributed to the social and engaging work environment during my time at the department. In particular, I want to thank Audun Gullikstad Hem for being my original office mate at Gløshaugen and later in the Nyhavna offices. In the end, we even got a collaborative project out of our discussions. Talking about the Nyhavna offices: Arriving there with the 'original crew' in the spring of 2021 was awesome. Thanks to everyone for numerous lengthy coffee breaks with hilarious off-topic discussions. They often made my day. A shout-out also goes to more recent arrivals who picked up the reins as the 'old guard' started graduating.

I would also like to thank Maritime Robotics and Eirik Moholt for facilitating our work at Brattørkaia by giving us access to a power outlet and bringing us coffee on several occasions. Oh, and let us not forget when our WiMAX antenna broke during the spring of 2022. You saved us weeks by bringing a reserve transceiver and some duct tape.

Finally, I would like to thank my parents, Eva and Ole. Without your support, I would surely not be where I am today. And for that, I am truly grateful.

March 2023, Trondheim
Petter Solnør

Contents

Acronyms

3DES Triple Data Encryption Standard

AES Advanced Encryption Standard
AES-NI AES New Instructions
ARP Address Resolution Protocol
ARX Add-Rotate-XOR
AUV autonomous underwater vehicle

CAN Controller Area Network
CBC Cipher Block Chaining
CCM Counter with CBC-MAC
CFB Cipher Feedback
CIA confidentiality, integrity, and availability
CRC cyclic redundancy check

DES Data Encryption Standard
DoS Denial of Service
DUNE Dynamic Unified Navigation Environment
DVL doppler velocity log

EO electro-optical

GCM Galois/Counter Mode
GNC guidance, navigation, and control
GNSS global navigation satellite system

HMAC Keyed-Hash Message Authentication Code

ICT information and communication technology
IDS intrusion detection system
ILOS Integral Line-of-Sight

IMC Inter-Module Communication
IMU inertial measurement unit
INS inertial navigation system
IP Internet Protocol
IV initialization vector

JPEG Joint Photographic Experts Group

LOS Line-of-Sight
LSTS Underwater Systems and Technology Laboratory

MAC Message Authentication Code
MD5 Message Digest 5
MiTM Man-in-The-Middle
MPC model predictive control

NCS networked control system
NED North-East-Down
NMEA National Marine Electronics Association

OCB Offset Codebook

PNG Portable Network Graphics
PTP Precision Time Protocol

ROS Robot Operating System
RPM revolutions per minute
RTK real-time kinematic

SHA Secure Hash Algorithm

TCP Transmission Control Protocol
TOV Time of Validity

UAV unmanned aerial vehicle
UDP User Datagram Protocol
USV unmanned surface vehicle
UTC Coordinated Universal Time

XOR exclusive-or

Chapter 1

Introduction

The topic of this thesis is the application of cryptographic methods in feedback control and autonomous systems. This first chapter contextualizes the work by providing the background and motivation pertaining to the work described in the thesis. To this end, the chapter also presents the thesis's structure by highlighting the contents and main contributions of coming chapters and the publications in which they are described.

1.1 Motivation

A significant benefit of industrialization and the modernization of society is the automation of manual labor. By taking advantage of advanced sensing technology, high-performance computing devices, and state-of-the-art telecommunications technology, modern algorithms provide unprecedented capabilities for situational awareness, decision & control, and collaborative problem-solving. In turn, these advances enable the automation of tasks of ever-increasing complexity in unpredictable environments, leading to increased productivity and prosperity. For example, increased automation enhances safety by reducing human exposure to dangerous environments and by limiting the risk of human error, a significant cause of accidents today [1].

Automation comes in many forms, from technical tools aiding in manual labor to fully automated processes operating without human oversight and intervention. At the most basic level, automation assists by solving individual tasks, such as maintaining a desired speed using automatic cruise control or keeping a desired course or heading using autopilots. More advanced automation procedures, such as adaptive cruise control and automatic lane-keeping in the automotive industry, typically require greater situational awareness and semantic understanding of the environment. Higher-level autonomy, that is, systems capable of perception and autonomous decision-making without human intervention, such as autonomous driving capabilities in cars and ferries, is also a very active field of research. In turn, these advances provide cost-efficient solutions in numerous industries, such as automatic inspection of power lines, underwater surveying and mapping, industrial manufacturing, public transportation, shipping, and remote surveillance [2].

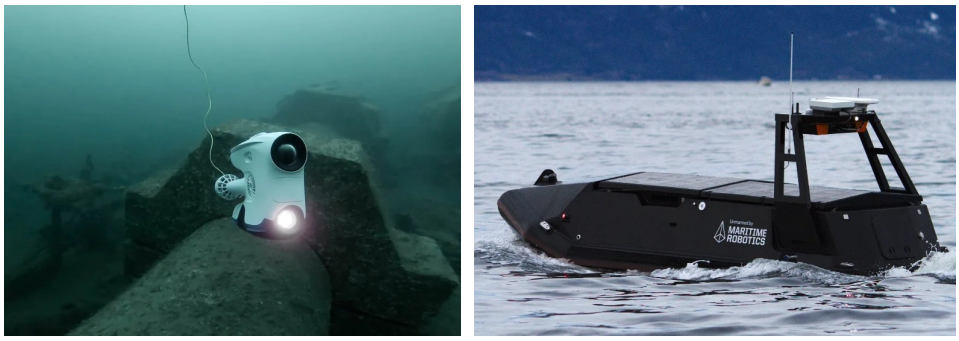
(a) (b)

Figure 1.1 (a) Autonomous ferry concept for urban public transport. Credit: Zeabuz. (b) Connected and intelligent autonomous ships, as envisioned by Rolls Royce. Credit: Rolls Royce.

For example, autonomous ferries, as shown in Fig. 1.1a, are being developed as a cost-efficient and non-invasive means of public transportation in urban environments. Moreover, there is significant potential for automation of commercial shipping, where seafarers, who are low in supply, spend most of their time in open waters, where automation is likely to succeed [3]. Autonomous vessels interacting and negotiating with other ships and with on-shore control centers, as illustrated by Fig. 1.1b, can also significantly increase safety at sea by reducing the number of people involved and by mitigating risks related to human error [1]. But automation is also advantageous on a smaller scale; instead of divers performing dangerous inspections of subsea installations at great depths, we can use underwater drones equipped with advanced sensing technology, as shown in Fig. 1.2a. Moreover, instead of using large manned vessels to conduct mapping and surveying operations, we can use smaller, unmanned surface vehicles (USVs), increasing cost-efficiency and lowering environmental impact, as shown in Fig. 1.2b. USVs can also be used for efficient short-distance freight operations as recently illustrated by the world's first fully autonomous freight route, initiated by Maritime Robotics across the Trondheim Fjord [4].

To a great extent, these developments are possible thanks to modern information and communication technology (ICT). In particular, increasing connectivity driven by recent advances in telecommunication technology provides high bandwidth and low latency connectivity between computing devices allowing wired and wireless real-time data-sharing and enabling collaborative systems. We refer to control systems whose sensors, actuators, and controllers are spatially distributed and connected over multi-purpose networks as networked control systems (NCSs). By offering benefits over centralized systems, such as ease of installation, flexible hardware- and software architectures, and reduced maintenance costs, NCSs have become widely adopted in numerous areas, such as sensor networks, automated highway systems, and unmanned aerial vehicles (UAVs) [5].

However, this increased connectivity comes at a cost. Cyber-physical attacks are a growing threat against numerous feedback control applications, including industrial processing units, chemical plants, and motion control systems [6, 7]. First,



(a) (b)

Figure 1.2 (a) The Blueye Pro underwater drone for subsea inspection. Credit: Blueye Robotics. (b) The Mariner USV is used for maritime data acquisition and autonomous transportation of goods. Credit: Maritime Robotics.

these systems can be vulnerable to industrial espionage through eavesdropping attacks, which, in turn, may pose a risk to intellectual property with significant financial ramifications and, in some instances, even threaten national security. Second, active attacks in these domains are particularly hazardous because the control systems manipulate underlying physical processes. By leveraging information obtained through eavesdropping attacks, malicious actors can mount active attacks to predictably change the behavior of these systems, putting equipment out of service or serving as attack vectors as part of terrorist attacks resulting in the loss of lives. Therefore, securing communication between the components of NCSs is of utmost importance [8].

These concerns are also relevant to the autonomous ships market, already valued as a multi-billion dollar industry and expected to face significant growth in the coming years [9]. Unfortunately, cybersecurity concerns threaten growth as cyberattacks pose a considerable risk, such as hijacking attempts of autonomous ships to steal goods or to use the ships as weapons in terrorist attacks [10]. For example, a hijacked vessel may target other vessels or off-shore and coastal installations such as cruise ships, oil & gas installations, and on-shore facilities, threatening human lives and causing dire financial consequences [11]. Consequently, several challenges remain before authorities, classification societies, and the general public are ready to accept the deployment of fully autonomous ships.

Industrial actors are also adopting cloud-computing technology to offload parts of the computation to software hosted on the hardware of third-party cloud providers, complicating matters further [12]. Such architectures introduce new challenges concerning the cybersecurity of control systems, particularly related to privacy. Intuitively, any information processed by the cloud server is vulnerable since, in most instances, it will have to be stored in unencrypted form on the hardware of the third-party provider. Verifying that tasks are executed correctly by the cloud server is, in most instances, relatively easy because the end user may possess knowledge of the expected output. Hence, we can detect deviations from operational procedures.

Moreover, if deviations are detected, such revelations becoming public knowledge would pose a significant risk to the business model of the cloud host. However, discovering intentional or unintentional disclosure of confidential system information is much harder. Therefore, we want to design privacy-preserving systems where such information leaks cannot occur; that is, we keep the benefit of using cloud-computing technology to outsource the computations without exposing confidential system information to inadvertent disclosure.

We usually describe security goals using the confidentiality, integrity, and availability (CIA) triad. Confidentiality corresponds to not disclosing confidential data to unauthorized elements, while integrity means ensuring the data is trustworthy and complete. Finally, availability means that units with the required authorization can reliably access the data upon request. For example, when we consider the transmission of data across insecure transmission channels in an NCS, the adversaries may be interested in both breaching the confidentiality of the transmitted data and manipulating its contents, that is, breaching the integrity. Concerning remote computation and cloud computing, and in light of the discussion above, we consider scenarios where data may leak, intentionally or unintentionally, from the cloud. That is, the cloud may breach the confidentiality of data passing through the cloud but does not actively seek to manipulate its contents. We refer to the latter type of adversary as *honest-but-curious* [13].

In this thesis, we consider both types of situations. We want to explore computationally efficient cryptographic methods of securing high-bandwidth applications across insecure transmission channels without inducing intolerable computational delays in the feedback control loop. Moreover, we want to explore techniques to ensure data privacy while outsourcing computations in control systems. Hence, the topics covered by the thesis are broad, with the common thread being applications of cryptography in feedback control to achieve secure data transmission across insecure communication channels or outsourcing of computations to third parties without adversely affecting privacy.

1.2 Background

There is a considerable body of existing work on conventional cybersecurity in feedback control systems and secure offloading of computational tasks. To put the contents of this thesis in a broader context, we provide a brief background by describing related advancements in the subjects covered. In each chapter, we will give a more detailed backdrop to the individual topics of the work described. Since the thesis consists of two parts addressing conventional cybersecurity and cryptography and privacy-preserving outsourcing of computations, we also divide the background accordingly.

1.2.1 Networked control systems and conventional security

From dedicated communication lines and busses, spatially distributed control systems are increasingly connected over general multi-purpose networks in the form of NCSs. The benefits of connecting system components over an existing

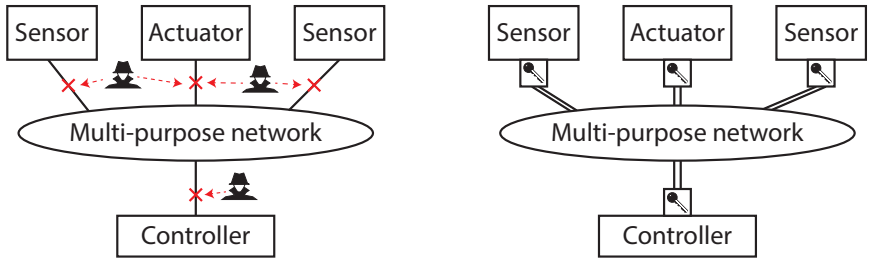


Figure 1.3 (a) A conventional NCS with vulnerable transmission channels. (b) An NCS with secure transmission channels.

network include lowered maintenance- and installation costs and more flexible system architectures than systems with dedicated communication channels [5]. Because of these benefits, NCSs are also increasingly used in vehicular systems [14, 15, 16]. However, the increasing connectivity associated with NCSs also poses new challenges related to cybersecurity. In particular, since control systems usually control some physical process, the potential damage inflicted by cyberattacks is severe. For example, an attacker can destabilize a chemical plant or manipulate vehicular control systems by injecting false data and commands through insecure network interfaces [7, 17], as shown in Fig. 1.3a.

The cybersecurity of NCSs has been a topic of concern for researchers for several years [6]. More recently, researchers have also raised concerns about the cybersecurity of vehicles equipped with NCSs and the need to secure the onboard communication channels against attacks [18, 19, 20]. Through numerous surveys and review papers, researchers have described attack surfaces and potential threats against vehicles with NCSs. For example, El-Rewini et al. [16] used a hierarchical framework to isolate threats and attacks against vehicular networks into categories concerning sensing, communication, and control. Regarding autonomous cars, Sun et al. [21] considered attack vectors against both inter- and intravehicular communication systems. If we look at the maritime domain, Silverajan et al. [19] described potential attack vectors against autonomous ships, which Kavallieratos et al. [22] later analyzed and classified using the STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege) approach. Specifically looking at intravehicular communication, Yağdereli et al. [23] considered attacks against onboard communication systems, such as eavesdropping, masquerading, and message modification attacks. However, the attack descriptions in these surveys and review papers are superficial, and few studies implement and demonstrate attacks in practice. Among practical demonstrations, Kang et al. [24] implemented an eavesdropping and spoofing attack against a Controller Area Network (CAN) bus in a conventional car. Concerning demonstrations of attacks in the maritime domain, Lund et al. [25] demonstrated a spoofing attack where they injected false global navigation satellite system (GNSS) positioning information. However, the spoofed positioning information was not used in closed-loop control. Hence, the viability of executing the attacks described in many of the surveys and review papers, and the

consequence of the spoofed signals in closed-loop control, remains unclear.

There are two primary methods of detecting and preventing cyberattacks; intrusion detection systems (IDSs) and cryptography. Considerable research has been devoted to developing various intrusion detection systems for NCSs, often motivated by claims stating that encryption and authentication mechanisms are resource-intensive or induce intolerable computational latencies [26, 27]. Moreover, IDSs can be retrofitted in a non-intrusive manner to monitor existing data traffic. However, the use of IDSs also poses significant practical challenges. Essentially, there are two types of IDSs; signature-based systems and anomaly detection systems. However, signature-based IDSs are not necessarily appropriate for dynamical systems because it is not clear *what* constitutes an attack. Hence, defining an attack signature is not trivial. The result is that most IDSs used in NCSs are anomaly detection systems. But the underlying principle of any anomaly detection system is to define what constitutes *normal* behavior and then warn the supervisor whenever the system exhibits behavior falling outside this definition. But obtaining an exact definition of 'normal behavior' for a dynamical system is infeasible since it entails accurately modeling and predicting all behaviors resulting from thrust allocations and external disturbances, several factors of which are unknown. Hence, anomaly detection systems use approximations, but these approximations lead to approximation errors that, in turn, lead to high false-positive rates; the system warns the supervisor of abnormal behavior when not being subject to an attack [28]. Since the underlying probability of most dynamical systems being under attack, in most instances, is exceedingly low, these false-positive rates significantly reduce the *practical use* of such systems. This problem is essentially an instantiation of the *base-rate fallacy* [29]. For this reason, anomaly detection systems are rarely used in practice [28], despite receiving considerable attention in academic settings.

The other approach consists of various approaches using cryptographic methods. Secure transmission mechanisms, constructed using cryptographic algorithms, have been designed to secure the data transmission between the components of the NCS, as illustrated by Fig. 1.3b. Notably, these transmission mechanisms have used block ciphers such as the Data Encryption Standard (DES), Triple Data Encryption Standard (3DES), Blowfish, and the Advanced Encryption Standard (AES) [30, 31]. However, these block ciphers are computationally expensive, particularly on computational devices with reduced instruction sets. Therefore, using these ciphers on embedded devices may introduce non-trivial computational latencies. Since the computational latencies limit the frequency with which the system components can operate, it may result in performance degradation. In particular, these considerations are relevant when transmitting data types of considerable size, such as point clouds and images.

1.2.2 Privacy-preserving outsourcing of computational tasks

From NCSs, it is natural to consider remote processing and computation on third-party infrastructure. Such outsourcing can be desirable for several reasons, such as improving cost-effectiveness, ensuring easy scalability, and providing ease of maintenance [32]. Because of these benefits, cloud computing finds applications in several industrial sectors, such as smart grids [33] and robotics [34, 35]. However,

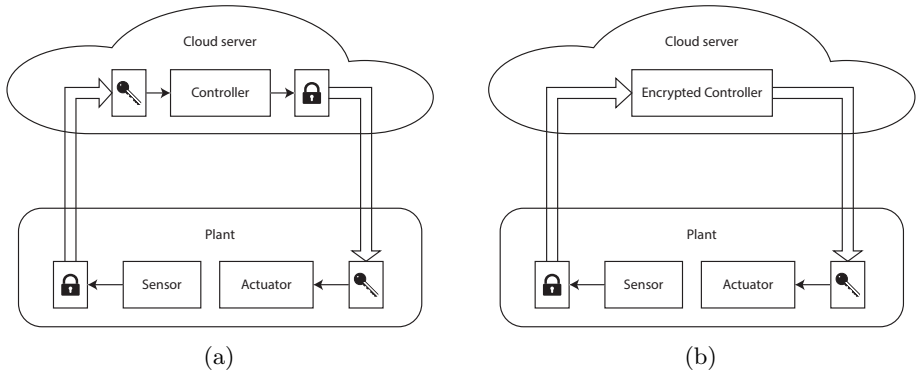


Figure 1.4 (a) A conventional control system hosted in the cloud. (b) An encrypted control system that operates directly on encrypted data hosted in the cloud.

outsourcing the computation to the cloud also introduces challenges concerning data privacy because a client may not entrust the cloud with its data. Hence, ever since Rivest et al. [36] introduced the concept of 'data banks' in the 1970s, it has been a goal for researchers to design techniques where we can outsource the computation while ensuring that the data remains private.

There are several approaches to achieving such privacy-preserving remote computation, most notably; secure multi-party computation, differential privacy, and homomorphic encryption. Secure multi-party computation denotes a set of schemes where a group of collaborating actors can evaluate a function in a distributed fashion without disclosing their private inputs [37]. To this end, secure multi-party computation has been used to achieve, for example, privacy-preserving information fusion [38] and control [39]. However, although we can use secure multi-party computation to evaluate a broad range of functions, it is generally interactive and requires several rounds of bidirectional communication between the participants. Hence, it is not particularly well suited for systems with real-time constraints [40].

Differential privacy is a different approach, initially used to compute statistics over a dataset without disclosing information about individual records [41]. Contrary to various *ad-hoc* 'anonymization' schemes, differential privacy provides a mathematically robust definition of privacy. Importantly, it is robust against *linkage attacks*, that is, attacks where the privacy of individual records in the 'anonymized' dataset can be lost if combined with *auxiliary information*, for example, from a second non-related database. There are several examples of such linkage attacks in practice, such as the attack against the Netflix dataset [42]. The working principle in differential privacy is to add an appropriate amount of noise to the data before allowing queries on the dataset. This noise is usually added by a trusted *curator* with access to the entire dataset while balancing the level of privacy against the accuracy of the statistics computed over the dataset.

Differential privacy was first introduced in the systems and control domain by Ny and Pappas [43]. In the same paper, they also described an architecture where the data providers add noise to the data directly without using a trusted curator.

This architecture is more appropriate for control applications and information fusion schemes that compute control actions and fuse information in real-time. However, the drawback of not using a trusted curator to add the noise to the dataset is that more noise must be added to obtain similar levels of privacy. This noise propagates through the computation and results in a more noisy output [43]. For certain applications, for example, information fusion schemes, accurate estimates are of utmost importance. Despite these drawbacks, differential privacy has been used to design private control systems and information fusion schemes, such as linear quadratic control [44, 45] and Kalman filtering [43, 46].

The last approach to achieving privacy-preserving computation, and the approach we are concerned with in Part II of this thesis, is homomorphic encryption. Homomorphic encryption denotes a set of cryptosystems where the encryption algorithm preserves some algebraic structure. Rivest et al. [36] first introduced the concept in the context of what they referred to as data banks. The idea was to use this algebraic structure to perform computations directly on encrypted data to produce a predictable result on the underlying unencrypted data. They referred to these encryption functions as *privacy homomorphisms* and the corresponding cryptosystems as *privacy transformations*. While these 'data banks' might have seemed somewhat contrived in 1978, they are pervasive today in the form of cloud servers.

Since then, homomorphic encryption has found several applications, such as enabling private queries in databases [47], machine learning on encrypted data [48], and electronic voting [49]. Early homomorphic cryptosystems, such as RSA [50], Elgamal [51] and Paillier [52] were only *partially homomorphic*; that is, the homomorphisms only hold for either addition or multiplication operations. Later, somewhat homomorphic cryptosystems that support a limited number of both homomorphic additions and homomorphic multiplications were proposed [53], and finally, in 2009, Craig Gentry proposed the first *fully* homomorphic cryptosystem [54]. Since then, several advances have been made, most notably concerning the computational efficiency of these cryptosystems. Nevertheless, fully homomorphic cryptosystems are computationally expensive [55], and as a result, applications resort to using partially and somewhat homomorphic cryptosystems whenever possible to maintain reasonable computational performance.

In the context of control systems, Kogiso and Fujita [56] first proposed using homomorphic encryption to design *encrypted control systems* that operate directly on encrypted data, as illustrated by Fig. 1.4b. Since then, the idea has gained significant traction and sparked an entirely new field of research [57]. Generally, we distinguish between *fully encrypted* and *semi-encrypted* control systems. The former corresponds to control systems where all control parameters and the data passing through the controller are encrypted. The latter corresponds to systems where this information is only partially encrypted, usually keeping the control parameter in plaintext form while the data passing through the controller is encrypted. To this end, homomorphic encryption has been used to design fully encrypted and semi-encrypted linear control [56, 58], semi-encrypted model predictive control (MPC) [59, 60, 61], and semi-encrypted cooperative control [62, 63]. Studies usually default to using the multiplicatively homomorphic Elgamal cryptosystem for fully encrypted control and the additively homomorphic Paillier cryptosystem for semi-encrypted

control systems.

Homomorphic encryption has also been used to achieve encrypted information fusion and filtering. The first example of encrypted information fusion was presented by Aristov et al. [64], who proposed an encrypted variant of the information filter, an algebraically equivalent variant of the Kalman filter. However, the method assumes that the system matrix is the identity matrix and the control matrix is null. Hence, it is not very practical. Later, Alexandru and Pappas [65] designed an encrypted linear quadratic Gaussian using a cryptosystem called labeled homomorphic encryption, while Zhang et al. [66] designed an encrypted state estimation scheme using a hybrid homomorphic encryption scheme. However, both designs are limited to static gain implementations. More recently, Ristic et al. [67] proposed an encrypted variant of *fast covariance intersection*, a method used to fuse unbiased Gaussian estimates. Because of its remarkable simplicity, the guarantee of consistency in the fused estimates, and computational efficiency, it is ubiquitous in industrial applications. However, the first encrypted variant was impractical when fusing many estimates. Moreover, it leaked the weights assigned to each input, revealing information about the performance of individual sensor systems. These problems were addressed in a new paper by the same authors [68], where they use an additively homomorphic cryptosystem to design an encrypted fast covariance intersection scheme.

Most research on the topic of encrypted control is devoted to theoretical aspects. For example, since the plaintext spaces of the cryptosystems used usually consist of integers, we need to map real-valued variables to valid elements of the plaintext space before encryption. This mapping consists of multiplication with a large integer, rounding, and a modulo operation. However, the multiplicative gain accumulates when we perform homomorphic multiplications. After a finite number of homomorphic multiplications, this accumulation causes an overflow in the plaintext space. Hence, stateful control systems, where the state of the controller is a function of the previous state, pose a significant problem and possible workarounds, for example, restricting system parameters to integers [69], periodically resetting the controller [70], and re-encrypting the state to remove the cumulative scaling factor [56], have been proposed. However, in practice, these approaches are often not viable. Therefore, the design of encrypted systems frequently boils down to simplifying a scheme such that computational constraints can be maintained while ensuring that the output produced is viable.

Research on practical applications of encrypted control is more limited [57]. For example, Tran et al. [71] demonstrated semi-encrypted control of an inverted pendulum implemented on a field-programmable gate array. Concerning motion control systems, there have been proof-of-concept demonstrations of encrypted vehicular control systems, for example, with semi-encrypted control of a wheeled robot [72] and a UAV [73]. However, these demonstrations were performed in controlled laboratory environments. Moreover, both demonstrations considered inner-loop control systems operating with high update rates. In practice, it would be more reasonable to outsource slower 'outer-loop' systems, such as path-planning and guidance systems.

1.3 Research objectives

In light of the background described above, the key objectives of this thesis are two-fold. The purpose of the first part of the thesis is to consider the conventional cybersecurity of feedback control systems connected over multi-purpose networks in autonomous vehicles. To increase the efficiency of secure signal transmission in such NCSs, we want to investigate the effect of using modern stream ciphers compared to conventional block ciphers. We hypothesize that these stream ciphers are more suitable to the instruction sets found in hardware used in feedback control systems. Therefore, using stream ciphers should reduce the computational latency induced. Moreover, contrary to previous work on the cybersecurity of NCSs and autonomous vehicles, we want to implement and demonstrate that NCSs are, in fact, vulnerable to practical cyber-physical attacks. Moreover, we also want to show that we can use such attacks to manipulate the path of autonomous vehicles.

In the second part of the thesis, we consider systems where we outsource part of the computation to an honest-but-curious cloud server. The goal is to design systems that ensure this does not leak confidential information to the server. To this end, we want to show that we can use homomorphic encryption to create an encrypted control system for a USV. Moreover, we want to show that we can use homomorphic encryption to implement encrypted guidance systems, that is, guidance systems that compute encrypted heading and course commands without knowing the vehicle's and the waypoints' location. To demonstrate the effectiveness of the encrypted control and guidance systems, we want to validate them through field experiments in an uncontrolled environment. Finally, we want to consider two methods of achieving accelerated and decentralized encrypted fusion of Gaussian estimates, respectively.

In summary, the main objectives of the thesis consist of

1. Investigating the benefit of using modern stream ciphers to secure the communication in NCSs.

2. Demontrating practical cyber-physical attacks against networked guidance, navigation, and control systems.

3. Considering the use of homomorphic encryption to design encrypted guidance and vehicular control systems.

4. Applications of cryptographic methods to achieve privacy-preserving fusion of unbiased Gaussian state estimates.

We highlight that throughout the thesis, we want to focus on a mix of new designs and proof-of-concepts validated through field demonstrations. Concerning implementations of encrypted systems, we focus on practical considerations such as the computational efficiency of the proposed methods. To this end, we also consider homomorphic cryptosystems not previously used to design encrypted control systems, to see whether we can improve the performance in terms of the computational latency induced.

1.4 Contribution and outline

The contents of the thesis are described in five journal papers, of which three have been accepted for publication, whilst two are in review at the time of this writing. Chapters 2 and 3 address conventional security in NCSs, while Chapters 4 and 5 treat applications of homomorphic encryption to design encrypted control and guidance systems for a USV, respectively. Then, in Chapter 6, we consider using stream ciphers to accelerate an encrypted information fusion scheme. Moreover, it introduces a decentralized variant constructed using a cryptographic concept called privacy-preserving aggregation. Chapter 7 concludes the thesis and also describes some directions that future work can pursue. Since the topics covered by the chapters vary quite a bit, we note that each chapter is self-contained. Hence, there are some notational differences between individual chapters, where each contains a section describing the notation used throughout that chapter. The following chapters are based on papers published in peer-reviewed scientific journals and papers undergoing peer review.

Part I: Cybersecurity and Symmetric Cryptography in Feedback Control

Chapter 2: Secure and Efficient Transmission of Vision-Based Feedback Control Signals

Publication:

[74] Ø. Volden, P. Solnør, S. Petrovic, and T. I. Fossen, "Secure and Efficient Transmission of Vision-Based Feedback Control Signals," *Journal of Intelligent & Robotic Systems*, vol. 103, pp. 1–16, Oct. 2021

Topic: An ever-increasing number of autonomous vehicles use bandwidth-greedy sensors such as cameras and LiDARs to sense and act to the world around us. Unfortunately, signal transmission in vehicles is vulnerable to passive and active cyber-physical attacks that may result in loss of intellectual property, or worse yet, the loss of control of a vehicle, potentially causing great harm. Therefore, it is important to investigate efficient cryptographic methods to secure signal transmission in such vehicles against outside threats. This study is motivated by the observation that previous publications have suggested legacy algorithms, which are either inefficient or insecure for vision-based signals.

Contribution: We show how stream ciphers and proper authenticated encryption can be applied to transfer sensor data securely and efficiently between computing devices suitable for distributed guidance, navigation, and control systems. We provide an efficient and flexible pipeline of cryptographic operations on image and point cloud data in the Robot Operating System (ROS). We also demonstrate how image data can be compressed to reduce the amount of data to be encrypted, transmitted, and decrypted. Experiments on embedded computers verify that modern

software cryptographic algorithms perform very well on large sensor data. Hence, the introduction of such algorithms should enhance security without significantly compromising the overall performance.

Chapter 3: Hijacking of Unmanned Surface Vehicles: A Demonstration of Attacks and Countermeasures in the Field

Publication:

[75] P. Solnør, Ø. Volden, K. Gryte, S. Petrovic, and T. I. Fossen, "Hijacking of unmanned surface vehicles: A demonstration of attacks and countermeasures in the field," *Journal of Field Robotics*, vol. 39, pp. 631–649, Aug. 2022

Topic: Driven by advances in ICTs, an increasing number of industries embrace unmanned and autonomous vehicles for services such as public transportation, shipping, mapping, and remote surveillance. Unfortunately, these vehicles are vulnerable to passive and active cyber-physical attacks that can be used for industrial espionage and hijacking attempts. Since attackers can use hijacked vehicles as weapons in terrorist attacks, ensuring secure operation of such vehicles is critical to prevent the attacks from causing dire financial consequences, or worse, the loss of human lives. This study is motivated by the observation that most cybersecurity studies provide superficial, high-level descriptions of vulnerabilities and attacks, and the true impact of the described attacks remains unclear.

Contribution: We demonstrate advanced manipulation attacks against an underactuated USV which results in successful hijackings. Using cryptography, we also show how the signal transmission can be secured to avoid hijacking attempts actively steering the vehicle off course. Through field experiments, we demonstrate how the attacks affect the closed-loop guidance, navigation, and control system and how the proposed countermeasures prevent these attacks from being successful. Our study is unique in that we provide a complete description of the attacked USV and give a detailed analysis of how spoofed navigation estimates affect the closed-loop behavior of the underactuated USV.

Part II: Applications of Homomorphic Encryption

Chapter 4: Development and Experimental Validation of an Encrypted Control System for an Unmanned Surface Vehicle

Publication:

[76] P. Solnør, S. Petrovic, and T. I. Fossen, "Development and experimental validation of an encrypted control system for an unmanned surface vehicle," *Robotics and Autonomous Systems*, pp. 1–17, 2023. (Submitted)

Topic: Advances in ICTs have caused a surge in cloud computing services. Moreover, the use of cloud computing resources for the automatic control of processes and robots has gained increasing attention in recent years. However, hosting control systems on third-party cloud infrastructures exposes the system to eavesdropping attacks and industrial espionage. To prevent such attacks and ensure the privacy of the data stored on the cloud, we can use homomorphic encryption and translate the control laws to mathematical operations in ciphertext space.

Contribution: We describe how we can create fully encrypted control systems using a concept called labeled homomorphic encryption. We also highlight that it can be beneficial to use the Joye-Libert cryptosystem instead of the Paillier cryptosystem when requiring an additively homomorphic cryptosystem. We proceed with a case study where we develop an encrypted surge speed and yaw controller for a USV, where the controller produces encrypted thrust allocations based on encrypted control parameters and state estimates. Results indicate that the proposed controller outperforms encrypted controllers constructed using multiplicatively homomorphic cryptosystems in terms of computational latency, and we demonstrate its efficiency through an extensive field experiment where we close the feedback control loop over a mobile broadband connection.

Chapter 5: Towards Oblivious Guidance Systems for Autonomous Vehicles

Publication:

[77] P. Solnør, S. Petrovic, and T. I. Fossen, "Towards oblivious guidance systems for autonomous vehicles," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2023. (Accepted, in press)

Topic: With cloud-computing technology, we can outsource computations in guidance systems to third-party providers, increasing scalability and enabling guidance-as-a-service. However, by remotely hosting a conventional guidance system on third-party infrastructure, we leak information such as the path of the vehicle. Potential customers may consider these leaks a serious breach of confidentiality, which limits the practical use of such systems. Therefore, to make cloud-based guidance systems viable, we would like to design guidance systems that we can host remotely without revealing confidential information to the host.

Contribution: We show that we can linearize Line-of-Sight (LOS) and Integral Line-of-Sight (ILOS) guidance laws to design guidance laws that we can implement using additively homomorphic cryptosystems. These guidance laws operate directly on encrypted position measurements, encrypted waypoints, and the bearing between each waypoint by using homomorphic encryption, effectively preventing the cloud host from identifying the vehicle's position. We show that the proposed guidance laws are locally exponentially stable and that the induced computational latency is appropriate for the real-time guidance of autonomous vehicles. Moreover, we

show that using the Joye-Libert cryptosystem instead of the Paillier cryptosystem results in a considerably faster implementation in terms of the computational latency induced. Through field experiments, we demonstrate that an encrypted guidance system is practical and allows an unmanned surface vehicle to follow an encrypted path. The originality of this work lies in conceptualizing, designing, and experimentally validating an encrypted guidance system, unlike other studies that considered encrypted control systems.

Chapter 6: Revisiting Encrypted Fast Covariance Intersection

Publication:

[78] P. Solnør and A. G. Hem, "Revisiting encrypted fast covariance intersection," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–11, 2023. (Submitted)

Topic: Chapter 6 considers a concept called *encrypted fast covariance intersection*. Fast covariance intersection is a computationally efficient method of fusing unbiased Gaussian estimates. It is ubiquitous in industrial applications because it does not require knowledge of the correlation between the individual input estimates and ensures consistency in the fused estimates. The conceptual idea of *encrypted* fast covariance intersection is to implement a variant of fast covariance intersection that fuses encrypted unbiased Gaussian estimates directly without leaking the input estimates and how they are weighted and without revealing the resulting fused output estimate.

Contribution: We describe two new variants of encrypted fast covariance intersection. The first method is an *accelerated variant* that uses computationally efficient stream ciphers, resulting in an acceleration of approximately five to six orders of magnitude compared to the original algorithm. Moreover, the accelerated variant reduces the amount of data transmitted by approximately 99% for an instantiation with a 128-bit level of security. The second method is a *decentralized* variant that eliminates the need for a fusion center. Because we no longer rely on a fusion center, we no longer have to assume that the recipients of the fused estimates are honest.

Other contributions

In addition to the publications above, I have authored the following papers during my time at the department:

[79] P. Solnør, "A Cryptographic Toolbox for Feedback Control Systems," *Modeling, Identification and Control*, vol. 41, pp. 313–332, Dec. 2020

[80] M. Akdağ, P. Solnør, and T. A. Johansen, "Collaborative collision avoidance for maritime autonomous surface ships: A review," *Ocean Engineering*, vol. 250, p. 110920, Apr. 2022

The former describes a software toolbox developed during my Master's thesis, which has been extended to include several additional algorithms during the work described

in this thesis. The latter is a survey paper on collaborative collision avoidance algorithms for maritime autonomous surface ships to which I contributed significantly. However, the topic addressed is not directly relevant to the contents pertaining to this thesis. I have also contributed significantly to the software and hardware development and the experimental validation of the systems onboard the Cyberotter experimental test platform, described in more detail in Appendix A. Finally, I have assisted colleagues in the department with field experiments for other research projects.

Part I

Cybersecurity and Symmetric Cryptography in Feedback Control

Chapter 2

Secure and Efficient Transmission of Vision-Based Feedback Control Signals

This chapter is based on the publication

[74] Ø. Volden, P. Solnør, S. Petrovic, and T. I. Fossen, "Secure and Efficient Transmission of Vision-Based Feedback Control Signals," *Journal of Intelligent & Robotic Systems*, vol. 103, pp. 1–16, Oct. 2021

2.1 Introduction

Autonomy has gained increased traction in the maritime sector over the past decade. By promising reduced costs and improved safety, USVs, autonomous underwater vehicles (AUVs), and UAVs may revolutionize services such as data acquisition, mapping, and remote surveillance [2]. However, significant challenges remain before autonomous vehicles are ready to enter the commercial market. In particular, these vehicles must be secure and robust against the growing threat of cyber-physical attacks for wide-spread acceptance by authorities, classification societies, and the general public [20]. In this context, we investigate how we can use cryptographic algorithms to protect sensor data in time-critical applications. The goal is to secure autonomous vehicles against passive and active adversaries with access to the transmission lines, as shown in Fig. 2.1.

Autonomous vehicles are controlled by guidance, navigation, and control (GNC) systems that perform path planning, estimate position, velocities, and attitude, and compute appropriate control signals to execute, respectively [81]. Often, these systems are modular since each task is performed by separate computational devices. The communication between these components is often done through local area networks, to which sensor devices are also connected. Feedback control systems that close the loop through networks are commonly referred to as NCSs [8]. By providing ease of installation, reduced maintenance costs, and flexible architectures, NCSs show promising advantages over systems with independent, dedicated communication

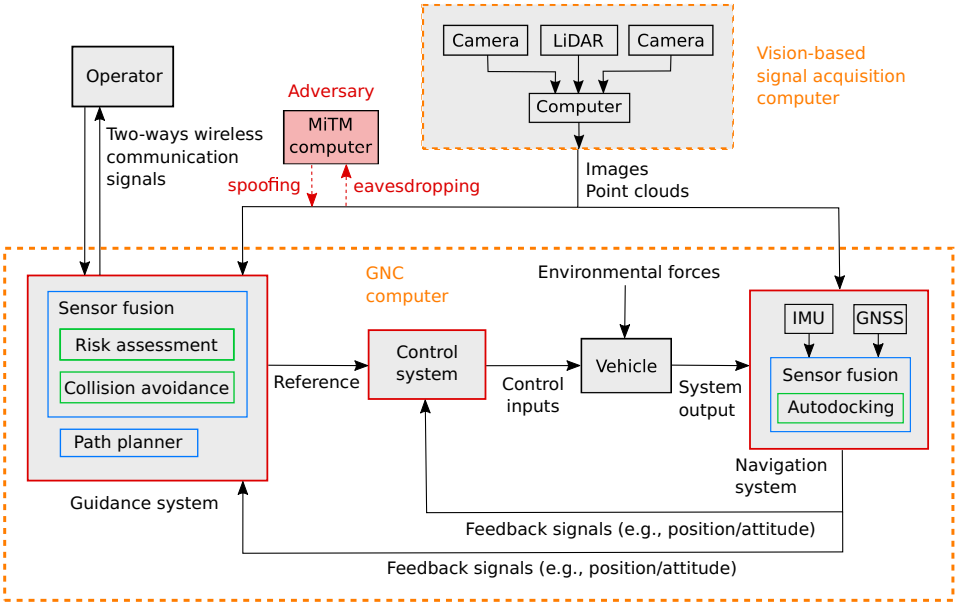


Figure 2.1 The figure shows a guidance, navigation, and control (GNC) system under attack by an adversary

channels [5].

For safety-critical tasks such as collision avoidance and autonomous docking, the GNC system must produce a detailed overview of the vehicle's environment. This is typically achieved with vision-based sensors such as cameras and LiDARs [82, 83, 84]. Because collision avoidance and autonomous docking are independent tasks performed by the guidance and navigation systems, respectively, they may require access to the same measurements. By using a dedicated computer for synchronized data acquisition, the raw data may be securely transmitted to the required systems, as seen in Fig. 2.1. Using local feature extraction to reduce the amount of data transmitted from the vision-based signal acquisition computer is possible. However, such a solution would blur the borders of the modular GNC design and increase system complexity. Alternatively, the data could be compressed before transmission, but compression may reduce the data quality and induce additional latency. It is therefore important to understand how compression algorithms affect the overall system performance.

The transmission of these signals, however, present attack surfaces which adversaries may exploit. Because of real-time requirements, the User Datagram Protocol (UDP) over the Internet Protocol (IP) is a common solution for signal transmission of large sensor data due to the simplicity of the protocols and the high bandwidth available [85]. Unfortunately, these protocols are inherently insecure and vulnerable to a number of cyber-physical attacks such as eavesdropping, bit manipulation, and packet injection by adversaries. If an adversary gains access to the signal transmissions in autonomous vehicles, he or she is free to eavesdrop on the data

and gain access to confidential system information. Furthermore, the adversary could inject its own data into the signal transmission to manipulate the behavior of the vehicle, as seen in Fig. 2.1. Such attacks may, for example, result in the failure of collision avoidance systems with dramatic consequences; rather than tracking incoming obstacles, the vehicle may be fooled into 'avoiding' objects that do not exist.

Traditionally, cryptography has been used to solve such problems. To prevent unauthorized parties from accessing the transmitted data, the data streams should be *encrypted*. To prevent the injection of spoofed data, the origin of the data streams should be *authenticated*. However, the large throughput required when processing images and point clouds may result in large time delays. This chapter aims to resolve this problem by demonstrating how state-of-the-art cryptographic algorithms result in considerably smaller time delays than existing works have suggested. We also seek to investigate whether compression before encryption can accelerate the cryptographic operations while also reducing the required throughput.

2.1.1 Related work

Since NCSs connect system components across a network, they become vulnerable to cyber-physical attacks such as eavesdropping and deception attacks, as described in [7, 17]. Therefore, the use of cryptographic algorithms such as the DES [86], 3DES [87], AES [88], Message Digest 5 (MD5) [89], and Keyed-Hash Message Authentication Code (HMAC) [90] has been suggested by [30, 91, 92] to secure the signal transmission in NCSs. In particular, Pang & Liu [91] suggested a 'secure transmission mechanism' consisting of a block cipher and a cryptographic hash function in a 'hash-then-encrypt' composition to provide security against deception attacks. However, the use of ad-hoc schemes to prevent deception attacks, such as the scheme proposed by Pang & Liu, has been shown to be cryptographically weak [93]. Instead, we argue that proper authenticated encryption, obtained through dedicated authenticated encryption algorithms or cryptographically strong compositions, for example, those investigated by Bellare & Namprempre [94], should be used.

Case-studies examining implementation-specific software such as ROS, for example, by Teixeira et al. [95], have discovered similar weaknesses to eavesdropping and data manipulation attacks. To counteract such attacks, well-known cryptographic algorithms such as 3DES, Blowfish [96], and AES have been applied in ROS [97, 98, 99]. Specifically, Rodrigues-Lera et al. [98] investigated the use of 3DES, AES, and Blowfish on images and LiDAR data in the ROS environment and found system performance to be adversely affected by the cryptographic operations. Due to the computational overhead, cryptographic algorithms induce latencies that are important to consider, especially if real-time requirements apply. Since the latency induced by cryptographic operations grows linearly with the data size, these latencies may be significant for large data such as images and point clouds.

The work described above examined the use of block ciphers accessed through open-source libraries. This leads Teixeira et al. to conclude that cryptography is not viable in all cases because the latency induced by encryption and decryption compromises real-time performance [95]. Interestingly, the use of state-of-the-art stream ciphers, such as those found in the eSTREAM portfolio [100] and the AEGIS

cipher [101], has not been considered by any of the previous work. Stream ciphers are stateful ciphers that usually consist of an *initialization phase* and a *keystream generation phase*. While the initialization phase does result in an initial overhead compared to block ciphers, the keystream generation phase of a stream cipher is much more efficient than that of a block cipher. This is important because vision-based signals are large compared to more common data transmitted in feedback control systems. For example, the transmission of a state estimate containing position, velocities, and attitude at a rate of 100 Hz would require a bandwidth around 10 KB/s. In comparison, images and point-clouds transmitted at 10 Hz would require bandwidth at least three orders of magnitude greater. Therefore, for encryption of vision-based signals, we suggest the use of dedicated stream ciphers, for which we expect to produce considerably better results in terms of latency.

2.1.2 Main contributions

The main objective of this study is to identify efficient cryptographic algorithms that avoid critical time delays in the feedback loop for autonomous vehicles. This is particularly important for vision-based signals processed by embedded systems onboard vehicles where the computational power is limited. To this end, we demonstrate that a stream of images and point clouds can be transmitted securely and efficiently between embedded systems by using modern cryptographic methods. We show how authenticated encryption can be used to obtain confidentiality, integrity, and data origin authenticity by combining stream ciphers and message authentication codes, or through a dedicated authenticated encryption algorithm. Finally, we demonstrate that by using the proposed algorithms, the use of compression before cryptographic operations results in larger time delays. Therefore, compression should only be applied if the network bandwidth is constrained. By suggesting the use of stream ciphers and authenticated encryption instead of block ciphers, this chapter presents an important contribution to the development of secure autonomous vehicles. Experimental results verify that the proposed algorithms significantly outperform algorithms suggested by others and should be used in autonomous vehicles.

The software-oriented stream ciphers from the eSTREAM portfolio are assessed and compared against the de facto standard, that is, the AES algorithm. The best-performing encryption algorithms are then composed with the HMAC algorithm to obtain authenticated encryption and then compared with the AEGIS algorithm. Finally, we investigate whether the use of compression before cryptographic operations results in increased performance. The algorithms have been integrated into the ROS environment as a cryptographically strong pipeline that enables cryptographic operations on image and point cloud data. The proposed pipeline is flexible since the data type is irrelevant to the implementation. Efficiency is ensured since the computational complexity is inherited from the underlying cryptographic algorithms and grows linearly with the data size. Source code, data set, and instructions to run the algorithms in ROS have been made available in a public Github repository [102]. The dataset is based on data collection onboard a USV and was created by the authors. To assess algorithm performance, experiments were conducted on edge-computing embedded devices rather than high-performance desktop computers.

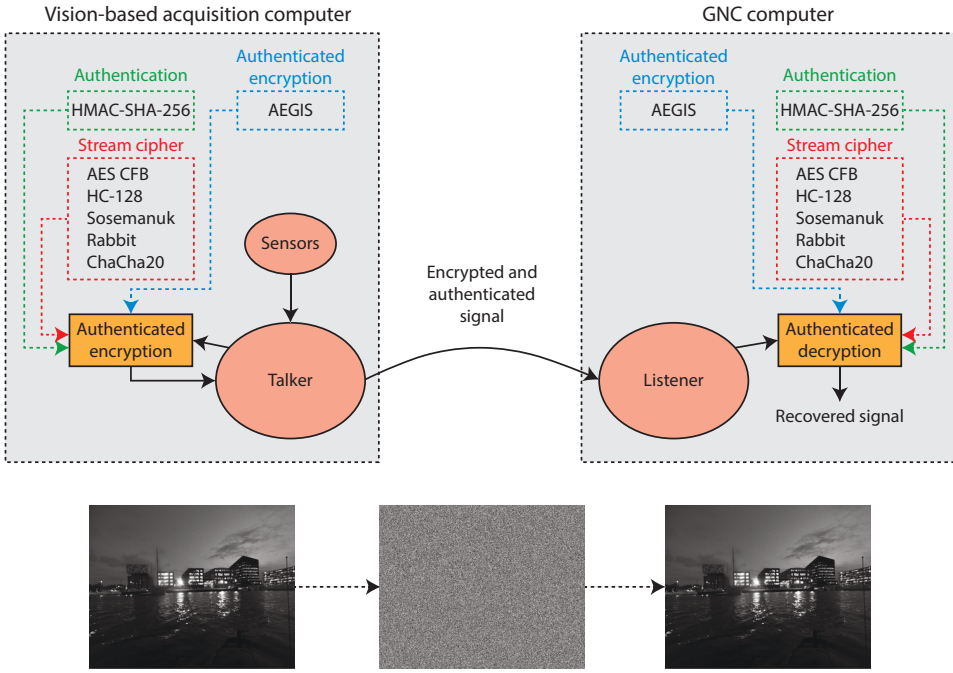


Figure 2.2 The block diagram shows the proposed cryptographic pipeline implemented in ROS.

Although the experiments are demonstrated for USVs, the embedded systems in use apply to a wide range of autonomous vehicles.

2.1.3 Outline

This chapter is structured as follows. Section 2.2 introduces the implemented methods and discusses how they are realized in ROS. Section 2.3 describes the hardware, software, and sensor data used in the experiments, the laboratory setup, and experiment-specific details. It also includes results and discussions regarding the experiments. Finally, we summarize the most important findings of this chapter and how they relate to the results achieved by other authors in Section 2.4.

2.2 Algorithms and implementation

First, we introduce the proposed cryptographic pipeline, followed by an introduction to ROS. Then, an overview of the relevant algorithms is given. Finally, we describe how the algorithms are integrated into the ROS environment. The focus lies on the communication between the vision-based signal acquisition computer and the guidance and navigation computers but applies to all signal transmissions seen in Fig. 2.1.

Introducing our analogy, used throughout this chapter, one of the computers acts as the *talker* while the other computer acts as the *listener*. The talker is the node that encrypts and transmits the sensor signals, while the listener is the node that receives and recovers the original message through decryption, as seen in Fig. 2.2. Authentication is included in the pipeline if the encryption algorithm does not provide data origin authenticity directly. That is, the talker node computes a tag based on the message before the message and tag are transmitted to the listener. Upon reception, the listener recomputes the tag and compares it with the received tag to validate the message's authenticity.

2.2.1 Communication and sensor interfacing in ROS

ROS is a flexible open-source framework for robot software development and is designed with distributed computing in mind. An essential part of ROS concerns how different computational processes can communicate and interchange messages. In this context, ROS *nodes* play an important role. In general, nodes are meant to operate at a fine-grained scale. Hence, robot control systems usually comprise several nodes. The use of multiple nodes also provides several benefits, such as reduced code complexity and additional fault tolerance as crashes are isolated to individual nodes. ROS *topics* are closely related to ROS nodes. Topics are named buses over which nodes exchange messages. ROS topics are intended for unidirectional streaming communication under the publisher and subscriber scheme. In general, nodes are not aware of to whom they are communicating. Instead, nodes subscribe to the topic containing data of interest generated by other nodes publishing data to the relevant topic. There can be multiple publishers and subscribers to a single topic. Furthermore, ROS supports both Transmission Control Protocol (TCP)/IP-based and UDP/IP-based message transport. The TCP/IP-based transport protocol, known as TCPROS, is the default communication protocol in ROS and streams data over persistent connections. The UDP/IP-based transport, known as UDPROS, is a low-latency, lossy alternative which separates messages into UDP packets. ROS nodes usually negotiate the desired transport at runtime. Nevertheless, it is possible to specify the choice of transport protocol manually.

This chapter focuses on large sensor data such as images and point clouds, both categorized under the *sensor_msgs* package in ROS. The cryptographic algorithms require that the data to be processed is *contiguous* in memory. Fortunately, data fields categorized under the *sensor_msgs* package are already serialized. We may therefore manipulate the data fields directly with cryptographic operations. After encryption, the encrypted data is represented as a byte stream which is easily embedded in a ROS topic through the *roscpp* library. The roscpp library enables C++ programmers to quickly interface with ROS topics and is designed to be the high-performance library for ROS.

2.2.2 Cryptographic algorithms

The cryptographic algorithms described in this chapter are *symmetric* in the sense that *a shared secret key* is used for encryption and decryption, and authentication and validation, respectively. The only *secret* part of the cryptographic algorithms is

Table 2.1 An overview of the cryptographic algorithms, the services they provide, and original work

Algorithm	Encryption	Authentication	Original work
AES	✓		[88]
HC-128	✓		[103]
ChaCha	✓		[104]
Rabbit	✓		[105]
Sosemanuk	✓		[106]
HMAC-SHA-256		✓	[90, 107]
AEGIS	✓	✓	[101]

material directly derived from the secret key, that is, the algorithms are entirely public. An encryption algorithm is considered *broken* if there exists a *reasonable* attack that is *more efficient* than a *brute-force attack*, that is, an exhaustive search through the key space. The input to the encryption algorithm is referred to as *plaintext*, while the output is referred to as *ciphertext*. By decrypting the ciphertext, the original plaintext is recovered as it was before encryption was applied.

Symmetric encryption algorithms are divided into two categories: block ciphers and stream ciphers. Block ciphers are stateless substitutions parametrized by a K-bit key operating on B-bit blocks. Examples are DES [86] and AES [88], which are well-known block ciphers. Since encryption of the same plaintext would always result in the same ciphertext, block ciphers are usually operated in specific *modes*, such as the Cipher Block Chaining (CBC) mode and Cipher Feedback (CFB) mode. When a block cipher is operated in the CBC mode it is still considered a block cipher, while the CFB mode converts the block cipher into a *stream cipher* by introducing a state. Stream ciphers work by extending a relatively short secret key into a much longer pseudorandom keystream which is then mixed with the plaintext, usually through the exclusive-or (XOR) operation, to form the ciphertext. Since stream ciphers are stateful, a unique, public parameter known as the initialization vector (IV) is mixed with the secret key to produce an initial state of the stream cipher before a message is encrypted or decrypted. In this sense, the IV serves as a cryptographic synchronization mechanism.

An overview of the relevant cryptographic algorithms that were integrated into the ROS environment and assessed can be seen in Table 2.1. The AES algorithm serves as a benchmark such that the performance of the algorithms may be compared to known results from related works. The algorithm implementations described in [79] were used. Brief descriptions of the algorithms are given. For additional details, the readers are advised to consult the corresponding references given throughout this chapter.

Advanced Encryption Standard

In 1997, the DES algorithm had been in place for over 20 years, and questions regarding the security of DES had haunted DES since its inception. Rather than

repeating the closed DES standardization process from the 1970s, the process of finding a new encryption standard, the AES, was decided to be public. Following a 3-year process, with careful examination of all the submissions concerning their security and performance, the Rijndael block cipher was selected in April 2000 and adopted as a federal information processing standard in early 2001 [88].

AES consists of multiple keyed rounds that operate on 128-bit blocks through a series of substitutions and permutations and quickly became the most popular encryption algorithm in-use. As a result of the widespread adoption of the standard, many processor architectures have implemented enhanced instruction sets, such as the *x86* Intel AES New Instructions (AES-NI) and *ARMv8* Cryptography Extension, that provide hardware acceleration of the AES operations.

eSTREAM Portfolio

During a discussion at the 2004 RSA Data Security Conference, the need for dedicated stream ciphers was questioned following the success of the AES. As an argument *for* the use of dedicated stream cipher designs, Shamir [100, page 1] identified two key areas in which dedicated stream ciphers could offer an advantage over block ciphers, namely: "*(1) where exceptionally high throughput is required in software and (2) where exceptionally low resource consumption is required in hardware.*"

As a result of the discussion, the ECRYPT Stream Cipher Project, a multi-year effort that ran from 2004-2008, was launched. The goal was to stimulate work on stream ciphers, and the project resulted in several successful entrants. The collection of successful entrants became known as the eSTREAM portfolio. The ciphers were designed to derive an initial state from a public IV and a secret key and to be optimized for software implementation (Profile 1) or hardware implementation (Profile 2) to address (1) and (2), respectively. Since we integrate the algorithms into the ROS environment, we focus on the stream ciphers optimized for software implementation.

The software-oriented portion of the eSTREAM portfolio consists of the following stream ciphers: HC-128 designed by Wu [103], Rabbit designed by Boesgaard et al. [105], ChaCha20 designed by Bernstein [104], and Sosemanuk designed by Berbain et al. [106]. We note that ChaCha20 is an improved version, with additional security margins and slightly increased performance, of the Salsa20 cipher, which is the original member of the eSTREAM portfolio. While a detailed description of each of the algorithms is out of scope, we mention that the algorithms differ structurally. These differences lead to the belief that if a weakness is identified for one cipher, the other ciphers are likely to remain unaffected. The HC-128 stream cipher uses large permutation tables, and the contents of the permutation tables determine the state. The Rabbit stream cipher uses elements from chaos theory, while the ChaCha20 stream cipher is an Add-Rotate-XOR (ARX) cipher. Finally, the Sosemanuk stream cipher uses a composition of a linear feedback shift register associated with a primitive feedback polynomial and parts of the Serpent block cipher, the runner-up submission to the AES competition.

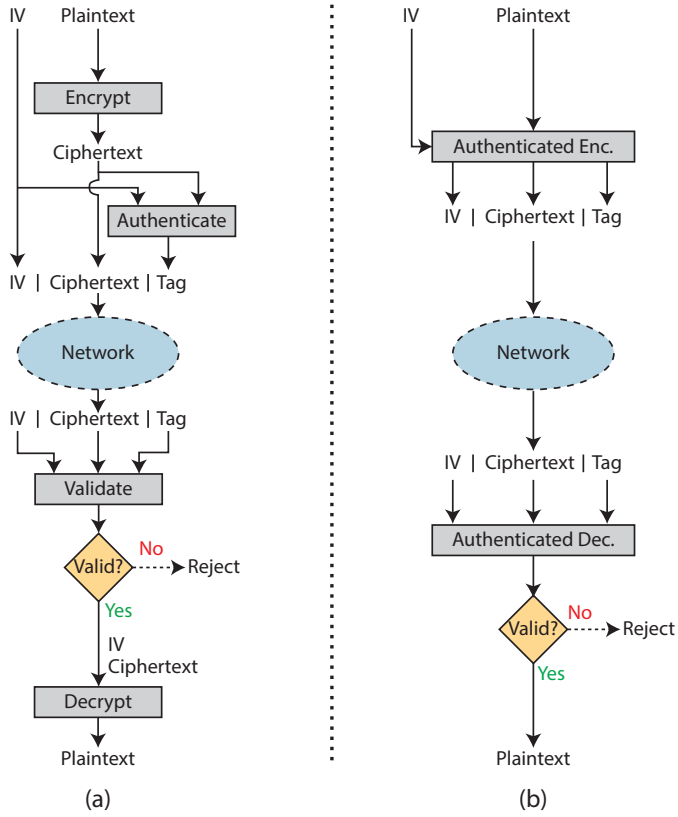


Figure 2.3 (a) Authenticated encryption is obtained through an 'Encrypt-then-MAC'-composition. (b) Authenticated encryption is obtained through a dedicated authenticated encryption algorithm such as AEGIS.

HMAC-SHA-256

Since encryption alone does not provide integrity nor data origin authenticity, a different cryptographic technique must be applied. A Message Authentication Code (MAC) is a symmetric-key construction which takes an arbitrary length input and produces a fixed B-bit output called a *tag*. A MAC algorithm is considered broken if an adversary is capable of forging a valid tag on *at least one message*. This is called an *existential forgery*.

The HMAC [90] is an algorithm that constructs a MAC from an unkeyed cryptographic hash function. In our experiments, we use the Secure Hash Algorithm (SHA) 256 (SHA-256) [107] as the cryptographic hash function per recommendation from the Internet Engineering Task Force [108]. By composing the HMAC algorithm with the encryption algorithms in the compositions described by [94], authenticated encryption is achieved. In our experiments, we use the 'Encrypt-then-MAC'-composition shown in Fig. 2.3a.

AEGIS

Instead of using a generic composition such as 'Encrypt-then-MAC', we may use a dedicated authenticated encryption algorithm, as seen in Fig. 2.3b. The Competition for Authenticated Encryption: Security, Applicability and Robustness (CAESAR) was an effort that ran from 2014-2019. The goal was to identify authenticated encryption schemes that provided better performance than those in use at the time, most notably AES Galois/Counter Mode (GCM) and AES Counter with CBC-MAC (CCM). The AEGIS cipher designed by Bart Preneel and Hongjun Wu was a successful entrant and is part of the high-performance portfolio along with AES Offset Codebook (OCB) mode designed by Rogaway et al. [109]. Unfortunately, AES OCB is encumbered by patents and is, therefore, not considered here. The AEGIS stream cipher was constructed with the AES round function in mind to take advantage of the AES-NI instruction set, but it also offers strong performance with table-driven variants of the AES round function and variants taking advantage of the ARMv8 Cryptography Extension. Since AEGIS provides authenticated encryption directly, there is no need to apply a separate MAC.

2.2.3 Compression algorithms

Contrary to encryption, compression may or may not lead to loss of information. Accordingly, we classify compression algorithms as *lossy* or *lossless*. A compression algorithm is lossy if it induces a loss of information, while a lossless compression algorithm permits perfect reconstruction of the original data. When evaluating the performance of compression algorithms, metrics such as *data compression ratio*, *data quality*, and *computational complexity* are relevant. The data compression ratio measures the relative reduction in size produced by the compression algorithm. Lossy compression algorithms, for example, Joint Photographic Experts Group (JPEG) [110], typically achieve high compression ratios (>10:1) without significantly reducing the data quality. Lossless compression algorithms such as Portable Network Graphics (PNG) [111] achieve much lower compression ratios, for example, up to 4:1 [112]. The computational complexity of lossy compression algorithms is typically a function of the data size. In contrast, the computational complexity of lossless compression algorithms also depends on the entropy of the data. Therefore, we expect lossy compression algorithms to operate in constant time for the given data size, while we expect the time complexity of lossless compression algorithms to vary.

2.2.4 Implementations in ROS

To implement authenticated encryption in the ROS environment, the data field of the ROS message must be altered to make space for the ciphertext, IV, and the message tag, respectively. This can be done by resizing the data field. Once resized, the IV is placed at the front of the data field belonging to the image or point cloud message, respectively. The plaintext is encrypted and authenticated using either an authenticated encryption cipher such as AEGIS or the 'Encrypt-

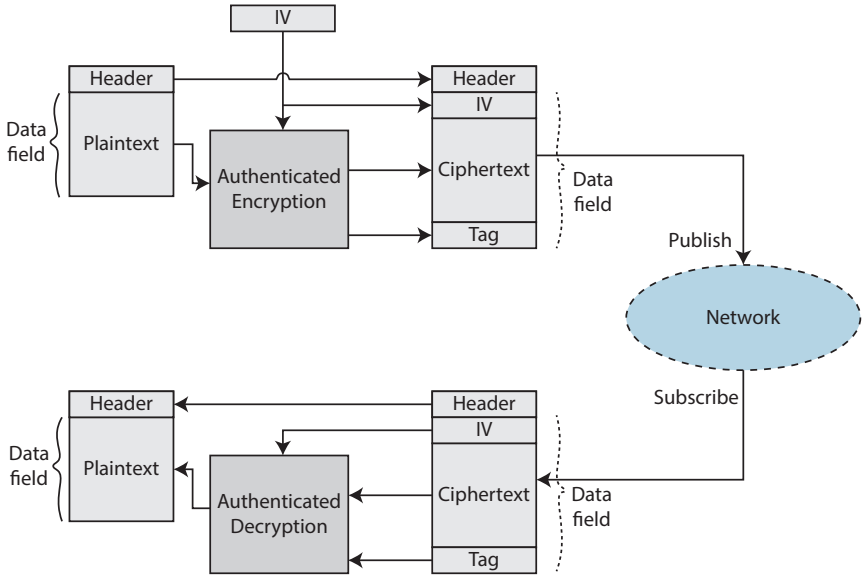


Figure 2.4 The figure shows how the original ROS message is secured through the use of authenticated encryption. The data field belonging to the encrypted ROS message also makes space for the IV and the message tag.

then-MAC' composition using a cipher and HMAC. As seen in Fig. 2.4, we only apply cryptographic operations on the data field belonging to the ROS message, thus leaving the header field unchanged before the complete ROS message is published to the ROS topic. Also, note that AEGIS and the 'Encrypt-then-MAC' composition rely on different approaches to obtain authenticated encryption. Hence, they will differ in the way they are implemented in ROS. The pseudocode for the authenticated encryption and the authenticated decryption nodes when an 'Encrypt-then-MAC'-scheme is used can be seen in Algorithms 1 and 2, respectively.

We implement image compression using lossless PNG and lossy JPEG compression, respectively. The compression algorithms are accessed through encoding/decoding functions in the OpenCV library. We use the ROS package *cv_bridge* to bridge ROS and OpenCV image data and then embed the encoding/decoding functions into the cryptographic pipeline. When combining compression and authenticated encryption, the order of the services plays an important role. For example, an 'encrypt-then-compress' scheme will result in low compression ratios because the ciphertext is very similar to white noise. In this scheme, lossless compression can be used but will result in very low compression ratios since there is no redundancy in white noise. Also, note that we cannot use lossy compression in such a scheme since it induces a loss of information. Consequently, the decryption algorithm is not given access to the original ciphertext and is therefore incapable of providing meaningful decryption. Therefore, we must apply the compression algorithms *before* the cryptographic operations. For this reason, we adopt a 'compress-then-encrypt' scheme in which authenticated encryption is applied to the output of the compression algo-

rithm. We refer to the Github repository for any additional implementation-specific details [102].

Algorithm 1 Authenticated Encryption Node

K: Symmetric key; IV: Initialization Vector;
$E_{K,IV}$: Encryption function parametrized by K and IV;
MAC_K: Message Authentication Code parametrized by K

Initialize MAC_K
while true **do**
    Initialize $E_{K,IV}$
    Plaintext ← Subscribe(Sensor Message)
    Ciphertext ← $E_{K,IV}$(Plaintext)
    Tag ← MAC_K(IV, Ciphertext)
    Secure Sensor Message ← (IV —— Ciphertext —— Tag)
    Publish(Secure Sensor Message)
    Update IV
end while

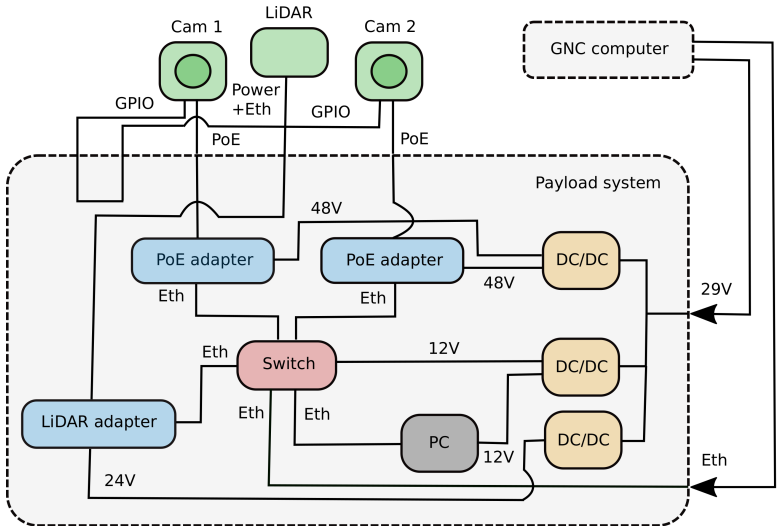
Algorithm 2 Authenticated Decryption Node

K: Symmetric key; IV: Initialization Vector;
$D_{K,IV}$: Decryption function parametrized by K and IV;
MAC_K: Message Authentication Code parametrized by K

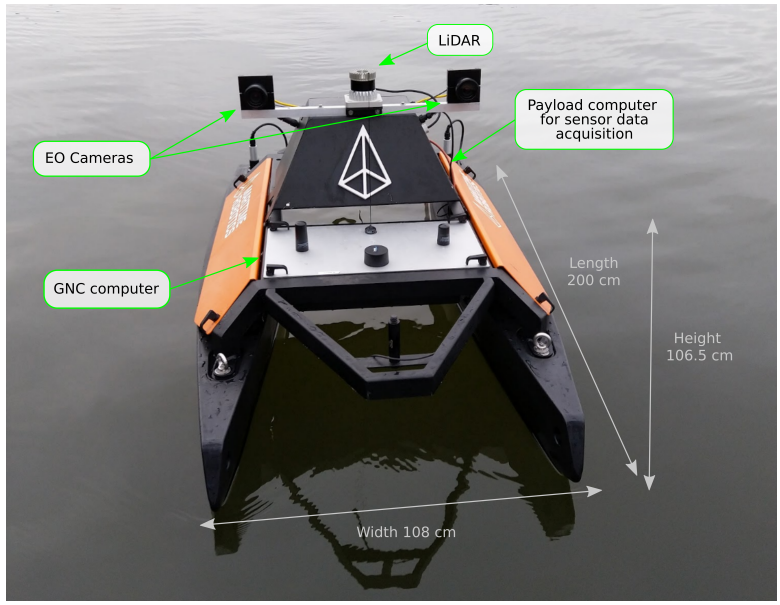
Initialize MAC_K
while true **do**
    (IV' —— Ciphertext' —— Tag') ← Subscribe(Secure Sensor Message')
    Tag ← MAC_K(IV', Ciphertext')
    if Tag != Tag' **then**
        Reject Ciphertext'
    else
        Initialize $D_{K,IV'}$
        Plaintext' ← $D_{K,IV'}$(Ciphertext')
        Sensor Message' ← Plaintext'
        Publish(Sensor Message')
    end if
end while

2.3 Experimental setup and testing of cryptographic algorithms

Following the description of the cryptographic algorithms, compression algorithms, and implementation-specific details, we move over to experiments. We begin by describing the hardware, software, and experimental data. Then, we describe how each experiment was conducted and the obtained results. Finally, we make some remarks regarding the obtained results.



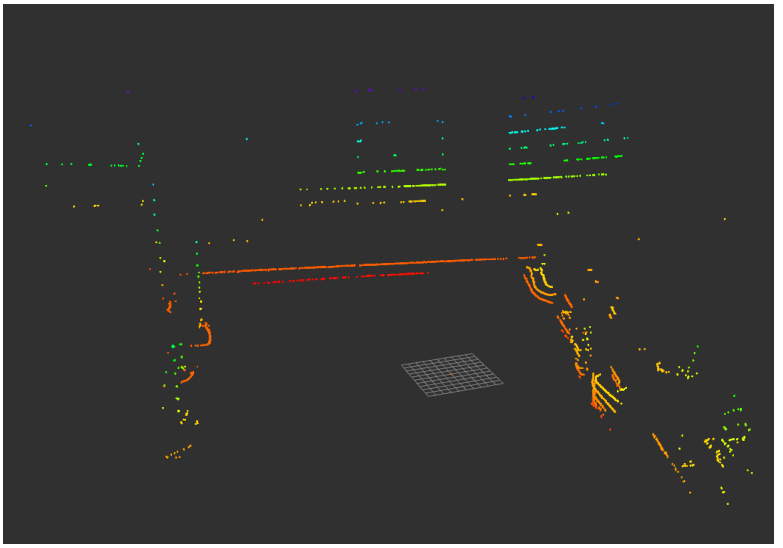
(a)



(b)

Figure 2.5 (a) The figure gives an overview of the hardware architecture. (b) The Otter USV from Maritime Robotics is armed with two electro-optical (EO) cameras and a LiDAR for sensor data acquisition. The image is reproduced with kind permission of Maritime Robotics, www.maritimerobotics.com.

(a)



(b)

Figure 2.6 The sensor data was recorded on-board a USV in the Trondheim Harbor. (a) shows a snapshot of the collected image data, and (b) shows a snapshot of the collected LiDAR data.

2.3.1 Hardware, software, and experimental data

We perform the experiments on an Nvidia Jetson Xavier Developer Kit. The Jetson Xavier is an efficient edge-computing unit with a small form factor, applicable for autonomous vehicles. It delivers 93.75 GB/s of highspeed I/O, which eases the burden of handling large amounts of data. Furthermore, the effect is adjustable between 10W and 30W, depending on the power mode. To utilize full performance, we set the power mode to "MAXN" to activate all eight cores. We use Ubuntu 18.04 LTS and the g++ 7.5.0 compiler. The software tools used are ROS Melodic and OpenCV 3.4.3.

Images and point clouds were acquired using two EO cameras and a LiDAR, respectively, onboard a small USV to test the algorithm performance on real-world data (see Fig. 2.5). The dataset contains image and point cloud data where the USV slowly navigates around the Trondheim Harbor (see Fig. 2.6). We use the Blackfly S GiGE camera with a 1280×1024 resolution. The images are recorded in monochrome pixel format at a frequency of 7.5 Hz. The LiDAR used is an Ouster OS-1 with a resolution of 2048×16 beams, running at 10 Hz. The data was recorded using the ROSbag tool to time stamp the data. The sensor data produced by the EO cameras and the LiDAR require a bandwidth of 9.75 MB/s and 31.5 MB/s, respectively. This data is then fed to the cryptographic pipeline to verify its value for real-world applications.

2.3.2 Experiments and results

The experimental setup consists of two Nvidia Jetson Xavier computers and one ethernet cable to enable wired, high-speed data transmission between the computing devices. To communicate and send data between nodes, ROS uses a master-slave setup. Only one node is assigned to be the master, and all other nodes must be configured to use this master. Using our analogy, the first computer, that is, the master node, acts as the talker, while the second computer, that is, the slave node, acts as the listener.

Experiment 1: Encryption latency measurements

In the first experiment, we are concerned with measuring the additional latency caused by encryption and decryption operations during the proposed cryptographic pipeline. We merge the latency caused by the encryption and decryption operations into one single cryptographic latency measurement, while network latency is ignored. Most of the computation is related to the core functionality accessed through initialization and processing. Minor parts relate to the allocation of input and output buffers and IV loading and incrementation for synchronization purposes. We refer to the public Github repository for additional details [102]. We benchmark AES in the CFB mode against the stream ciphers from the eSTREAM portfolio, that is, HC-128, Sosemanuk, ChaCha20, and Rabbit. Both a table-driven (software-oriented) and a hardware-accelerated (ARMv8 Cryptography Extension) variant of AES CFB is benchmarked.

Results

Table 2.2 summarizes the cryptographic latency measurements for the encryption-only schemes on image and point cloud data across 1000 consecutive samples. These samples were used to calculate the mean and standard deviation. A visual representation of the results is shown in Fig. 2.7. Observe that the relative order between the algorithms is the same when comparing image and point cloud data. As shown, Rabbit produces the lowest latency closely followed by HC-128, while AES CFB produces the highest latency. In between, we find ChaCha20, Sosemanuk, and the hardware-accelerated variant of AES CFB, respectively.

Experiment 2: Authenticated encryption latency measurements

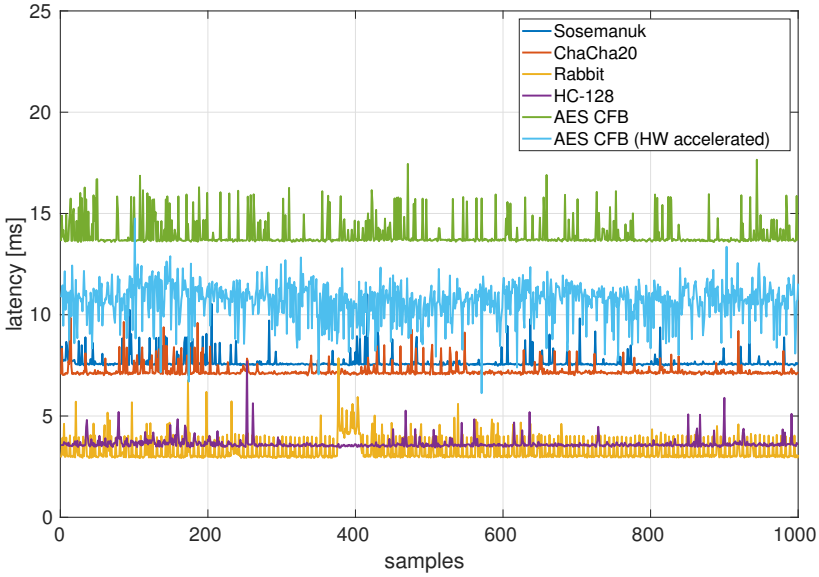
The encryption algorithms assessed in Experiment 1 provide confidentiality only and do not ensure the integrity and authenticity of the message upon reception. Therefore, for active attacks such as spoofing and message manipulation, we need to apply authenticated encryption. Consequently, we compare 'Encrypt-then-MAC' compositions with the authenticated encryption algorithm AEGIS in the second experiment. The 'Encrypt-then-MAC'-compositions consist of Rabbit and HC-128 composed with the HMAC-SHA-256 authentication algorithm, as they performed best in experiment 1. Both table-driven and hardware-accelerated variants of AEGIS are tested. As before, we merge the cryptographic operations into one single latency measurement.

Results

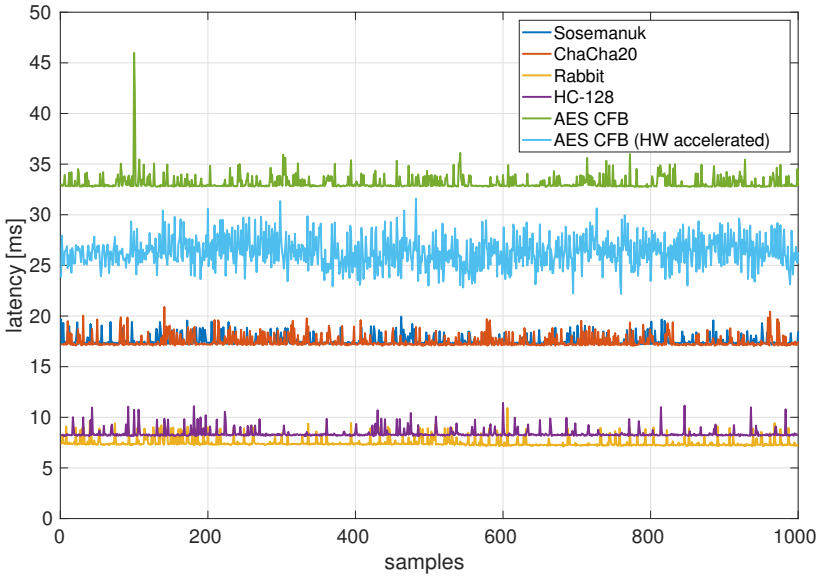
Table 2.3 summarizes the cryptographic latency measurements for the authenticated encryption schemes on image and point cloud data across 1000 consecutive samples. The samples were used to calculate the mean and standard deviation. A visual representation of the results is shown in Fig. 2.9. The hardware-accelerated variant of AEGIS proves to be the most efficient, followed by the table-driven AEGIS implementation, which is considerably faster than the HC-128+HMAC and Rabbit+HMAC schemes.

Experiment 3: Combining image compression and cryptographic operations

In the third experiment, we investigate whether compression before cryptographic operations is faster than cryptographic operations on the original data. Due to the remarkably efficient stream ciphers benchmarked so far, it is interesting to investigate if this is the case. Since we focus on authenticated encryption, and given the results from Experiment 2, AEGIS is used in a 'compress-then-encrypt' scheme. We assess the performance of 'compress-then-encrypt' schemes in which both lossy and lossless compression algorithms are composed with authenticated encryption. The lossy compression algorithm JPEG and the lossless compression algorithm PNG are used. The pipeline can be seen in Fig. 2.8.



(a)



(b)

Figure 2.7 (a) The figure shows the cryptographic latencies for image data across 1000 samples when encryption schemes are used in Experiment 1. (b) The figure shows the cryptographic latencies for point cloud data across 1000 samples when encryption schemes are used in Experiment 1.

Table 2.2 The table shows a latency comparison for various encryption algorithms over 1000 samples related to experiment 1. Original works for the algorithms are found in Table 2.1

Encryption algorithm	**Mean latency [ms]**	**Std.Deviation latency [ms]**
Image data size: 1.31 MB	μ	σ
Sosemanuk	8.0400	0.5935
ChaCha20	7.2184	0.3188
Rabbit	3.3055	0.5740
HC-128	3.9312	0.4858
AES CFB	14.6879	0.6558
AES CFB (HW accelerated)	10.6753	0.9205
Point cloud data size: 3.15 MB	μ	σ
Sosemanuk	18.6308	0.4184
ChaCha20	17.4546	0.5578
Rabbit	7.4948	0.4810
HC-128	9.3381	0.4608
AES CFB	33.3859	0.7673
AES CFB (HW accelerated)	26.3027	1.4332

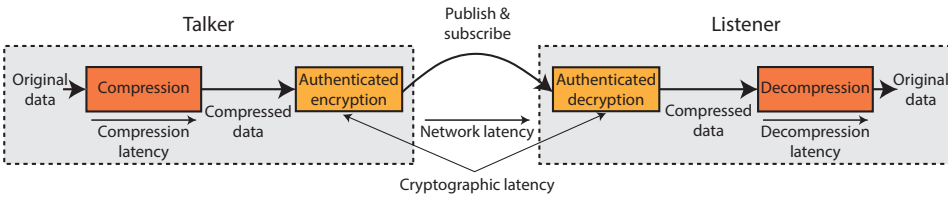
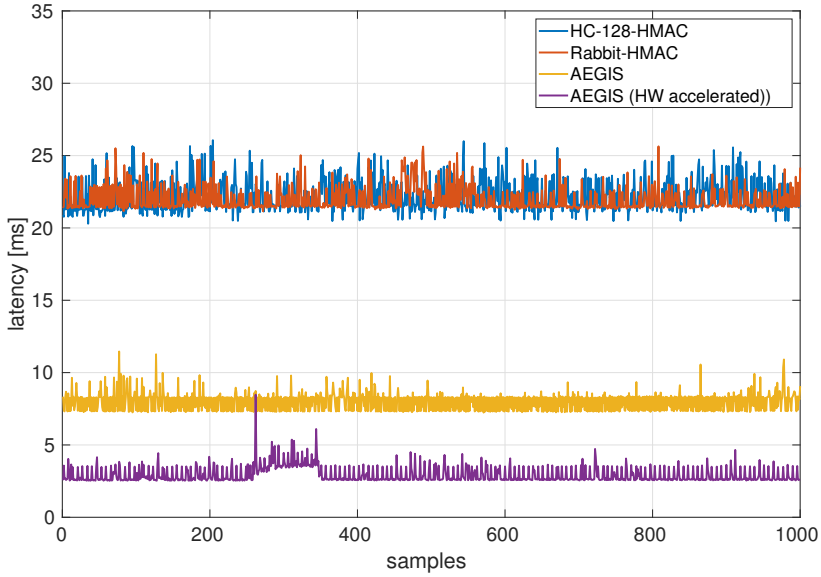


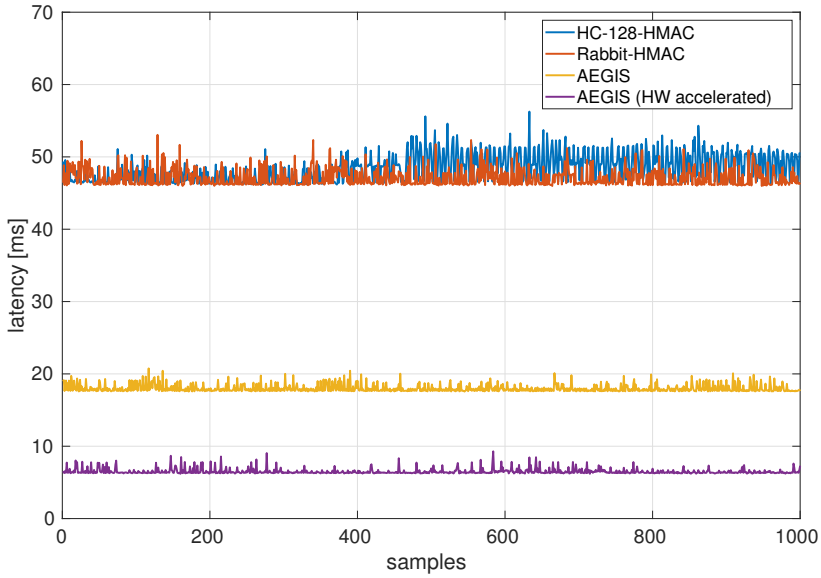
Figure 2.8 The third experiment investigates a 'compress-then-encrypt' scheme where compression and authenticated encryption are combined. The latencies related to authenticated encryption and decryption operations, respectively, are merged into one measurement, that is, the cryptographic latency.

Results

Table 2.4 summarizes the latencies based on image compression and cryptographic operations across 1000 consecutive samples. These samples were used to calculate the mean latency related to compression, cryptographic operations, and decompression, respectively. A visual representation of the results is shown in Fig. 2.10. On average, the original images were reduced by 74% and 30% when JPEG and PNG were used, respectively. Consequently, the cryptographic latency was reduced as well. However, this reduction is offset by the time it takes to perform compression and decompression. Note that lossless PNG compression and decompression produces significantly higher, and varying latencies than lossy JPEG, as expected.



(a)



(b)

Figure 2.9 (a) The figure shows the cryptographic latencies for image data across 1000 samples when encryption schemes are used in Experiment 2. (b) The figure shows the cryptographic latencies for point cloud data across 1000 samples when encryption schemes are used in Experiment 2.

Table 2.3 The table shows a latency comparison for authenticated encryption algorithms over 1000 samples related to experiment 2. Original works for the algorithms are found in Table 2.1

Auth. enc. algorithm	**Mean latency [ms]**	**Std.Dev. latency [ms]**
Image size: 1.31 MB	μ	σ
HC-128 + HMAC	22.3717	0.6925
Rabbit + HMAC	21.9876	0.8219
AEGIS	7.9323	0.6350
AEGIS (HW accelerated)	2.9235	0.5449
Point cloud size: 3.15 MB	μ	σ
HC-128 + HMAC	48.3779	1.8759
Rabbit + HMAC	47.1431	1.2136
AEGIS	17.9901	0.5358
AEGIS (HW accelerated)	6.4980	0.3858

Table 2.4 Latency comparison for PNG+AEGIS and JPEG+AEGIS in which both are benchmarked up against AEGIS on raw image data, that is, no compression, using 1000 samples. The original work of AEGIS is found in Table 2.1

Authenticated encryption algorithm	**Compression algorithm**	**Mean compressed data size [MB]**	**Mean latency compression [ms]**	**Mean cryptographic latency [ms]**	**Mean latency decompression [ms]**	**Mean total latency [ms]**
Image data size: 1.31 MB		n	μ_1	μ_2	μ_3	μ
AEGIS	JPEG[a]	0.31	11.5161	1.9274	6.7596	20.2037
AEGIS	PNG[b]	0.92	137.0573	5.3907	21.0929	163.5409
AEGIS	-	-	-	7.9323	-	7.9323

[a] JPEG compression level is set to default value according to the OpenCV specification [113], that is, 95/100.
[b] PNG compression level is set to default value according to the OpenCV specification [113], that is, 3/9.

2.3.3 Remarks

The results show that the stream ciphers significantly outperform the AES block cipher, and we recommend the Rabbit and HC-128 stream ciphers if only confidentiality is required. However, when used in generic compositions, they are outperformed by the authenticated encryption algorithm AEGIS. We believe this is because the AEGIS algorithm derives a message tag from the internal state and does not require the instantiation and initialization of a separate MAC, which is the case for the generic compositions. As such, we recommend that AEGIS is used when authenticated encryption is required.

The use of compression and decompression reduces the data size and, subsequently, the cryptographic latency. However, this gain is offset by the latency induced by compression and decompression, as seen in Experiment 3. The latency induced by the PNG compression was considerable, even at a relatively low compression ratio. While smaller compression ratios would result in reduced compression time, the size

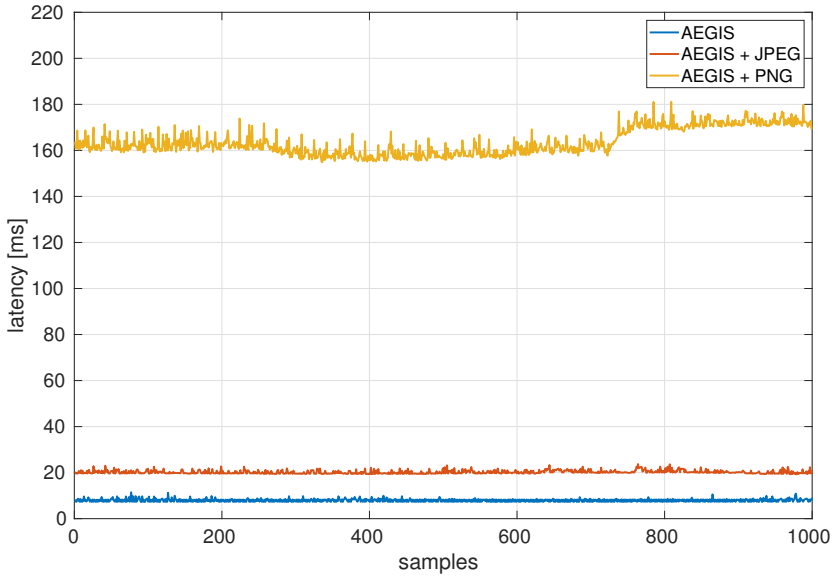


Figure 2.10 Overall latencies when image data is compressed, encrypted, decrypted, and decompressed across 1000 samples in Experiment 3. Here, lossless PNG and lossy JPEG compression are combined with AEGIS to reduce the amount of image data to be encrypted, transmitted, and decrypted.

of the compressed image rapidly converges to the size of the original image. As such, we find PNG compression to be unsuitable for time-critical applications. Regarding JPEG, we find that the total latency of JPEG + AEGIS is higher than if AEGIS is applied directly. However, JPEG also reduces the bandwidth required significantly and might therefore be considered if bandwidth is constrained. Interestingly, if less efficient cryptographic schemes are used, for example, AES CFB + HMAC, lossy compression may be beneficial. That is, the 'compress-then-encrypt' scheme may produce lower total latency than if cryptographic algorithms are applied directly. This is significant since it implies that the cryptographic algorithms proposed by previous work, for example, [97, 98, 99], could benefit from lossy compression algorithms. This is no longer the case when AEGIS is used.

Another aspect to consider is the data quality. Since the data used in guidance, navigation, and control must be of high quality, that is, near-lossless, we set the compression ratio low. Due to the low compression ratio, the Huffman encoding step of the JPEG algorithm must process a relatively large amount of data with $O(n \log n)$ run time [114]. This step is believed to be the bottleneck of the pipeline. With higher compression ratios, the JPEG algorithm is faster but never faster than using AEGIS directly. Additionally, increasing compression ratios will progressively deteriorate the data and thereby the performance of the GNC system.

2.4 Chapter summary

With the increased use of vision-based sensors such as cameras and LiDARs in autonomous vehicles, it is essential to consider how the signals from these sensors can be secured efficiently. The vision-based signals pose a significant challenge because they require much greater throughput than traditional signals in feedback control. Previous research on how vision-based feedback control signals can be secured has been restricted to the use of block ciphers, which are much less efficient.

We address this problem by suggesting modern stream ciphers and demonstrate that these ciphers perform much better than the block ciphers proposed by previous work. We have also demonstrated that AEGIS gives the best results if authenticated encryption is required. Finally, we find that while compression may accelerate the cryptographic pipeline for algorithms proposed by other authors, this is no longer the case when AEGIS is used. As a result, compression algorithms should only be combined with AEGIS if network bandwidth is constrained. All algorithms have been implemented in ROS and made publicly available through a Github repository [102]. In the future, we plan to implement and conduct full-scale experiments to show that the proposed method indeed applies to more resource-demanding feedback control applications.

Chapter 3

Hijacking of Unmanned Surface Vehicles: A Demonstration of Attacks and Countermeasures in the Field

This chapter is based on the publication

[75] P. Solnør, Ø. Volden, K. Gryte, S. Petrovic, and T. I. Fossen, "Hijacking of unmanned surface vehicles: A demonstration of attacks and countermeasures in the field," *Journal of Field Robotics*, vol. 39, pp. 631–649, Aug. 2022

3.1 Introduction

With an increased focus on autonomy, the autonomous ships market has become a multi-billion dollar industry and is expected to face significant growth in the coming years [9]. Leveraging advanced ICT, unmanned and autonomous vehicles have shown significant advantages for services such as public transportation, environmental monitoring, mapping, and remote surveillance and are predicted to play an essential role in the future [2]. Industrial leaders are racing to develop advanced autonomous solutions for ferries [115] and cargo ships [116], respectively. Additionally, commercialization of ideas from public research projects, for example, the Autoferry project [117], occurs through spin-off companies seeking to develop autonomous ferries for urban public transportation.

Unfortunately, cybersecurity concerns threaten the growth of the autonomous ships market [10]. Hijacking attacks of autonomous ships pose a crucial threat, as they may be used for stealing goods or as weapons in terrorist attacks. Targeting other vessels or off-shore and coastal installations, for example, cruise ships, oil & gas platforms, and on-shore centers, such attacks threaten the lives of civilians and may cause dire financial consequences [11]. Consequently, several challenges remain before fully autonomous ships can be accepted by authorities, classification societies, and the general public.

At the core of autonomous vehicles are advanced GNC systems [81]. Often implemented as distributed systems, the GNC components communicate over buses and networks spanning the vehicle. Historically, CAN buses have been used for this purpose; however, Ethernet is becoming an increasingly popular option for intra-vehicular communication [14, 15]. Generally, we refer to feedback control systems closing the loop over networks as NCSs [8]. With the ease of installation and reduced maintenance costs due to flexible software and hardware architectures, NCSs provide significant advantages over systems with independent communication channels [5]. Nevertheless, these communication lines are inherently insecure, making NCSs vulnerable to cyber-physical attacks. Additionally, developers often use middleware frameworks such as ROS and the Underwater Systems and Technology Laboratory (LSTS) toolchain [118] to implement NCSs. In fact, according to a study by ABI Research [119], ROS is expected to be present in a large fraction of future commercial robotic systems. However, these frameworks do not provide additional security mechanisms, and researchers have expressed concerns about the security of these frameworks for some time [95, 120]. Therefore, it is essential to address these vulnerabilities, and as such, ROS 2, currently under development, includes additional security mechanisms [121].

While researchers have expressed concerns over the security of intra-vehicular communication, attacks taking advantage of the vulnerabilities are rarely demonstrated. This may lead to a false sense of security among system developers when using popular software frameworks. As a result, in this chapter, we describe and demonstrate how we can exploit these vulnerabilities to hijack and take control of an underactuated USV, thus bridging the gap between theory and practice. We also demonstrate how we can prevent these attacks by securing the GNC communication with modern cryptographic algorithms. The experiments are performed on the Cyberotter USV shown in Fig. 3.1.

3.1.1 Related work

Because of the great benefits associated with NCSs, they are increasingly used in vehicles [16]. However, since NCSs connect system components across a network and are vulnerable to cyber-physical attacks such as eavesdropping and data injection [7, 17], researchers have expressed concerns about the cybersecurity of NCSs for many years [6]. In particular, with increased self-governance, security breaches in onboard communication systems may directly cause altered behavior in unmanned and autonomous vehicles. As such, there is a growing concern about the cyber-physical resilience of these vehicles [18, 19, 20], and it is therefore critical to establish secure communication between the connected devices.

Numerous surveys and review papers have described vulnerabilities and cyber-attacks against vehicles. El-Rewini et al. [16] describe vehicular cybersecurity challenges using a hierarchical framework to isolate threats and attacks in three layers; sensing, communication, and control. Considering a broad scope of attack vectors against inter and intra-vehicular communication, Sun et al. [21] discuss cybersecurity vulnerabilities related to autonomous cars. Similarly, in the maritime domain, Silverajan et al. [19] describe relevant attack surfaces for unmanned smart ships. These attack surfaces, and cyber-attacks against autonomous ships, were

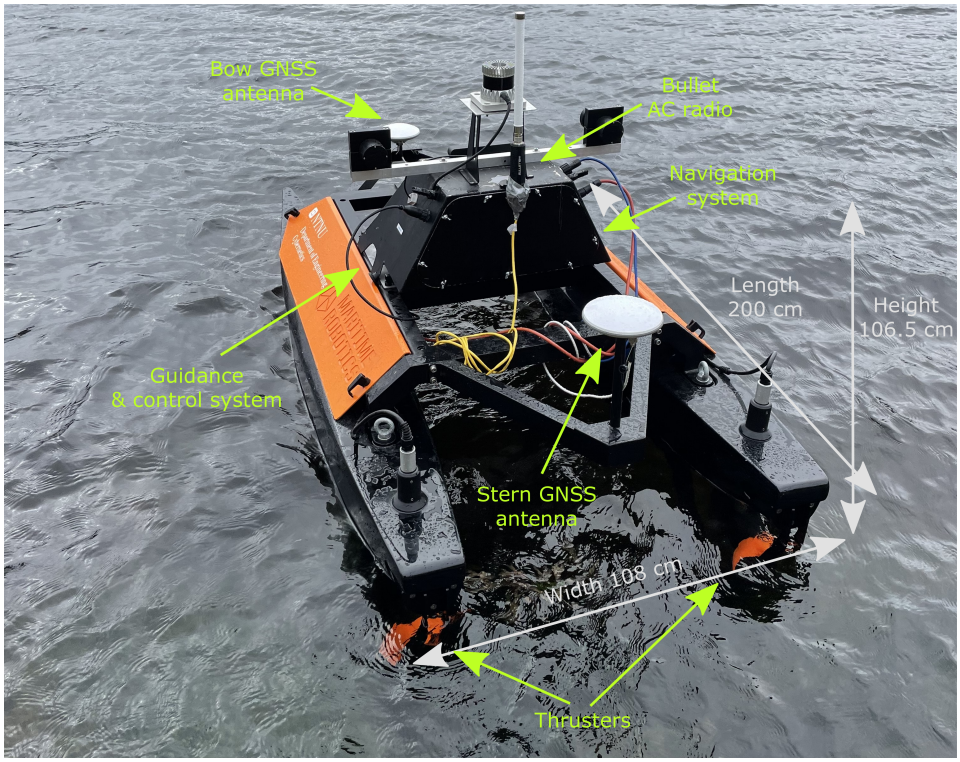


Figure 3.1 An overview of the Cyberotter USV.

later analyzed and classified according to the STRIDE approach by Kavallieratos et al. [22]. With a focus on intra-vehicular communication, Yağdereli et al. [23] describe attacks targeting communication lines, such as passive eavesdropping and active masquerading and message modification attacks, against unmanned and autonomous vehicles. Notably, these studies provide superficial descriptions, and the viability of executing the attacks, and the resulting consequences, remain unclear.

Considering cyber-attack demonstrations on intra-vehicular communication, Kang et al. [24] implemented an attack against a CAN bus in a conventional car, where messages were first eavesdropped upon and analyzed, followed by the injection of spoofed messages. In the maritime domain, Lund et al. [25] demonstrated an attack on an integrated inertial navigation system (INS) solution, where the estimated position of the vessel was changed by spoofing National Marine Electronics Association (NMEA) messages coming from the GNSS receiver. While the CAN bus attack was implemented in a controlled laboratory setup, the INS attack was performed in the field with results visible on an Electronic Chart Display and Information System. Nevertheless, conventional, manned vehicles were the target of both attacks. Hence, the signals are not used directly in closed-loop control. As such, we find that the literature lacks studies demonstrating how unmanned and autonomous vehicles with increased self-governance are affected by such attacks.

To detect cyber-attacks against intra-vehicular communication, we can use cryptographic methods or anomaly-based IDSs. The use of anomaly-based IDSs is often motivated by claims stating that encryption and authentication methods conflict with the link layer data frames used or are too resource-intensive [26, 27]. However, these assumptions may be problematic in many practical applications. Firstly, anomaly detection methods are problematic themselves because they require accurate definitions of normality. This is very challenging, causing anomaly detection methods to suffer from high false-positive rates [28]. A high false-positive rate, combined with a low probability of attack, that is, *base rate*, is problematic because of the base rate fallacy phenomenon [29]. For this reason, anomaly-based IDSs are rarely used in practice [28]. Secondly, regarding the use of cryptographic algorithms, we argue that the cryptographic algorithms rarely have to be used at the link layer. Just like cryptographic operations are not applied on the payload of Ethernet frames, they need not be applied on the payload of CAN bus frames. Instead, they can often be used higher up in the communication protocol stack, for example, at the application layer. Concerning the efficiency of cryptographic algorithms, we find that modern, symmetric cryptographic algorithms are very efficient and can, therefore, be applied to feedback control systems without inducing significant time delays [74]. For example, Mun et al. [122] have suggested using cryptographic authentication methods on a CAN bus and conducted laboratory experiments for validation that demonstrated their efficiency.

3.1.2 Main contributions

Rather than reiterating high-level descriptions of cyber-physical attacks and related countermeasures, the main objective of this study is to demonstrate that cyber-physical attacks can indeed be implemented and used to hijack a USV. We also show that our proposed cryptographic methods can prevent these attacks from being successful. In particular, we describe how manipulation of yaw (that is, heading) and position estimates changes the behavior of an underactuated USV. We proceed by describing how these attacks can be implemented and then suggest countermeasures that secure the GNC communication against eavesdropping, injection, and replay attacks. Finally, we implement the attacks on the insecure and the secured system and conduct field experiments to verify that the attacks are indeed successful in hijacking the vehicle without cryptographic protection and that the cryptographic methods successfully detect and prevent such attacks. The proposed cryptographic methods are beneficial compared to previously proposed anomaly-based IDSs because the problems of high false-positive rates and false-negative rates are reduced to a minimum if symmetric cryptographic algorithms are used. Consequently, contrary to anomaly-based IDSs, the proposed methods are appropriate for practical applications. In summary, the following are considered the main contributions of this study:

- We describe and analyze how manipulation of position and heading estimates affect the closed-loop behavior of an underactuated USV.
- We provide a detailed description of how these attacks can be implemented.

- We describe how cryptographic methods can prevent such attacks and argue that they are more practical than previously proposed anomaly-based IDSs.
- We implement and demonstrate the effect of the described attacks and defensive measures on a USV.

3.1.3 Outline

The remainder of this chapter is structured as follows. In Section 3.2, we introduce cryptographic concepts relevant to securing distributed GNC systems. We then present the case study in Section 3.3, where we introduce USV motion control. Based on this, we describe how eavesdropping and spoofing attacks can be used to manipulate the USV and how cryptographic measures can prevent these attacks. In Section 3.4, we show the experimental setup and describe the experiments. Then, in Section 3.5, we describe and discuss the experimental results. Finally, Section 3.6 concludes the chapter.

3.2 Cryptography

When a USV uses a distributed GNC system, it becomes vulnerable to cyber-attacks if adversaries gain access to the transmission lines. In fact, the usual assumption in security analysis is that adversaries do have access to the transmission lines. For example, an adversary with such access can eavesdrop on the communication to obtain confidential information or inject spoofed messages to manipulate the behavior of the USV. Such attacks may be used for industrial espionage and hijacking purposes. By using cryptographic methods, we can prevent these attacks from being successful.

3.2.1 Cryptographic concepts and terminology

Cryptography is typically used to achieve secure signal transmission (confidentiality) across insecure communication lines. Today, in the analysis and design of cryptographic algorithms, it is assumed that the cryptographic algorithm is known by the adversary, and only the keys, and material directly derived from the keys, are kept secret. This is commonly referred to as *Kerckhoff's Principle*.

Symmetric and asymmetric cryptography Cryptographic schemes are classified as *symmetric* and *asymmetric*, depending on whether the transmitter and the receiver use the same keys or not. Asymmetric cryptographic schemes are often based on number-theoretic problems that are believed to be hard, such as finding the prime factorization $p, q \in \mathbb{N}$ of a composite number $N = p \cdot q \in \mathbb{N}$, where p and q are of approximately the same size (in bits), or finding the discrete logarithm b of a group element $a = g^b \in \mathbb{G}$ given the very large group $\mathbb{G}$, the group element a, and the generator of the group g. On the other hand, symmetric cryptographic schemes are built using finite state automata, bitwise operators such as *AND*, *OR*, and *XOR*, and transpositions and highly nonlinear substitutions. Consequently, symmetric cryptography is much faster than asymmetric cryptography in software. However,

asymmetric cryptography brings other unique properties, such as the possibility of non-repudiation and symmetric key exchange. Since the GNC components are assumed to be trusted entities and key exchange is not required, these properties are unnecessary. As such, we will only consider symmetric cryptography in this chapter.

Encryption Encryption algorithms are used to obtain confidential signal transmission over insecure transmission channels. We refer to an encryption algorithm as a *block cipher* or a *stream cipher* depending on whether the algorithm is *stateless* or *stateful*. While block ciphers are N-bit substitutions parameterized by a secret K-bit key, the stream ciphers work by extending the key to a much longer pseudo-random sequence known as the *keystream*. Since the encryption algorithms need to work across insecure transmission channels, the stateful stream ciphers require a cryptographic synchronization mechanism. This is typically achieved using a public parameter known as the IV. The IV and the secret key are used to derive an initial state of the cipher, typically on a per-message basis. The input to an encryption algorithm is called *plaintext*, while the resulting output is called *ciphertext*. By decrypting the ciphertext, the corresponding plaintext is recovered. Without access to the secret key, the ciphertext should be computationally indistinguishable from white noise. An encryption algorithm is considered broken if an attack that recovers the key and/or the plaintext with computational complexity less than 2^K exists. Today, a keysize of 128 bits or more is recommended for data that needs to be protected after 2030 [123].

Authentication Unfortunately, encryption does not ensure the integrity nor confirmation of the true origin of the message, that is, asserting that information received is from a trusted source. This is referred to as *data origin authenticity*. Data origin authenticity may be obtained through the use of MACs. A MAC is a function parameterized by a secret, shared key that maps a message of arbitrary size to a fixed B-bit output. The output of the MAC is referred to as a *tag* and is transmitted with the message. Upon reception, the receiver, in possession of the secret key, re-computes the tag and compares the tag with the received tag. If the tags match, the message is considered authentic. In addition to resistance against key recovery attacks, a MAC should resist *existential forgery attacks*, that is, it should be infeasible for an adversary without knowledge of the secret key to produce a valid (message, tag)-pair for a new message. Assuming the MAC used is cryptographically secure, the computational complexity of an existential forgery is $2^{\frac{B}{2}}$ because of the *birthday attack* [124, p. 143]. Consequently, a tag size of 128 bits results in 64-bit security against existential forgery. The key size used in the MAC should be similar to that used in encryption algorithms, while the tag size depends on other considerations, such as the feasibility of testing large quantities of (message, tag)-pairs for the adversary. The most commonly used MAC is the HMAC, which constructs a MAC from cryptographic hash functions [90].

Authenticated encryption Since both confidentiality and data origin authenticity are desirable properties, encryption and MACs are often combined. This

is referred to as *authenticated encryption*. Authenticated encryption can be obtained through the use of *generic compositions* such as 'encrypt-then-MAC' [94] or through dedicated algorithms designed to provide both confidentiality and data origin authenticity directly, such as AEGIS [101].

3.2.2 Fault checks and cryptographic authenticity

Before continuing, we emphasize the difference between conventional *fault checks* and cryptographic MACs. Fault checks such as parity bits, checksums, cyclic redundancy checks (CRCs), and *hash codes* are public, *unkeyed* algorithms designed to detect *inadvertent transmission errors* or *data integrity breaches*. As such, anyone with knowledge of the specific fault check used can forge valid messages. This is fundamentally different from MACs, for which it should be computationally infeasible for an adversary to compute a valid (message, tag)-pair for a new message, that is, an existential forgery.

Communication protocols frequently use conventional fault checks to discard corrupted messages. However, the existence of such fault checks does not make the system secure against active adversaries. These adversaries can forge valid messages that the receiver accepts. Examples of frameworks that use conventional fault checks and not cryptographic MACs include the Inter-Module Communication (IMC) protocol, used in the LSTS toolchain. Other frameworks, such as ROS, do not even use conventional fault checks [120].

3.3 Case-study: Attacking and securing a USV

We proceed by introducing motion control systems for underactuated USVs. Based on this, we show how we can spoof the heading and the position to cause predictable changes in the paths of USVs, illustrating that both are means of hijacking. We proceed by describing the technical implementation of the spoofing attacks. Finally, we show how cryptographic methods can be used as countermeasures to prevent such attacks.

3.3.1 USV motion control

Let $\boldsymbol{\eta} = [N, E, \psi]^T \in \mathbb{R}^2 \times \mathbb{S}$ describe the vehicle pose and $\boldsymbol{\nu} = [u, v, r]^T \in \mathbb{R}^3$ describe the vehicle velocity in the earth-fixed North-East-Down (NED) reference frame and the body-fixed frame, respectively. To control the USV, a motion control system consisting of three independent system blocks, *guidance*, *navigation*, and *control*, is usually used. Notably, many USVs have two controls, for example, a propeller and a rudder. Consequently, these USVs can only directly control u and ψ, that is, surge speed and yaw, and are, therefore, underactuated. The navigation system estimates the position, velocity, and attitude of the USV, for example, by using GNSS receivers and an inertial measurement unit (IMU), and the guidance system uses these estimates and the desired path to compute the desired yaw and the desired surge speed of the USV. The control system then uses the estimates from the navigation system and the desired yaw and surge speed from the guidance

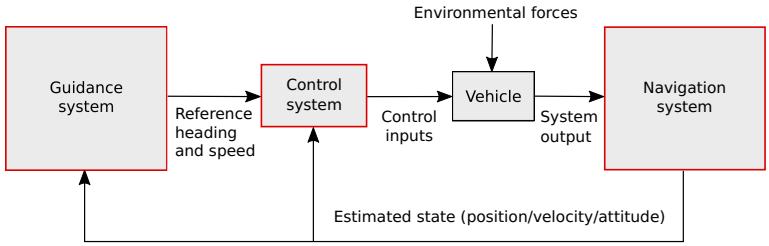


Figure 3.2 A generic motion control system for an underactuated USV.

system to allocate thrust to the actuators of the USV. The signal flow between the GNC components is shown in Fig. 3.2.

3.3.2 Vehicle manipulation

Underactuated USVs usually solve the path following problem by defining a 2-D workspace consisting of along-track and cross-track errors and then using a LOS guidance law to minimize the cross-track error [125, p. 258]. Let the variables $\psi, \hat{\psi}$, and ψ_d denote the true yaw, the estimated yaw, and the desired yaw of the vehicle, respectively. The true position of the USV is denoted by $p^n = [x,\ y]^T$, the estimated position of the USV is denoted by $\hat{p}^n = [\hat{x},\ \hat{y}]^T$, and we assume that the USV is following a straight-line path, implicitly defined by the two waypoints $p_k^n = [x_k,\ y_k]^T$ and $p_{k+1}^n = [x_{k+1},\ y_{k+1}]^T$. Moreover, we consider a path-fixed reference frame, rotated by a positive angle α_k relative to the x-axis of the NED frame, whose origin is located in p_k^n and whose x-axis is tangential to the path. The position of the USV in the path-fixed frame is computed as

$$[s,\ e]^T = R_n^p(\alpha_k)(p^n - p_k^n), \tag{3.1}$$

where $R_n^p(\alpha_k) \in SO(2)$ is a rotation matrix from the earth-fixed NED frame to the path-fixed frame. As such, the path-fixed s-coordinate describes the along-track distance, and the e-coordinate describes the cross-track error. Additional details are found in [125, p. 258].

The desired yaw is given by

$$\psi_d = \chi_d - \beta_c, \tag{3.2}$$

where χ_d is the desired course and $\beta_c = \text{atan2}(v, u) \in \mathbb{S} = (-\pi, \pi]$ is the crab angle caused by currents and wind. Assuming the crab angle is slowly varying, it can be handled with integral action and set to zero [126]. The desired yaw of the vehicle, assuming a lookahead-based LOS guidance system is used, is then given by

$$\psi_d = -\text{atan2}(e, s_\Delta), \tag{3.3}$$

where s_Δ denotes the look-ahead distance to an intersection point (x_{los}, y_{los}) on the desired path to p_{k+1}^n [126]. Assuming an adversary manages to spoof the yaw angle by an offset $\Delta\psi$, we have

$$\hat{\psi} = \psi + \Delta\psi, \tag{3.4}$$

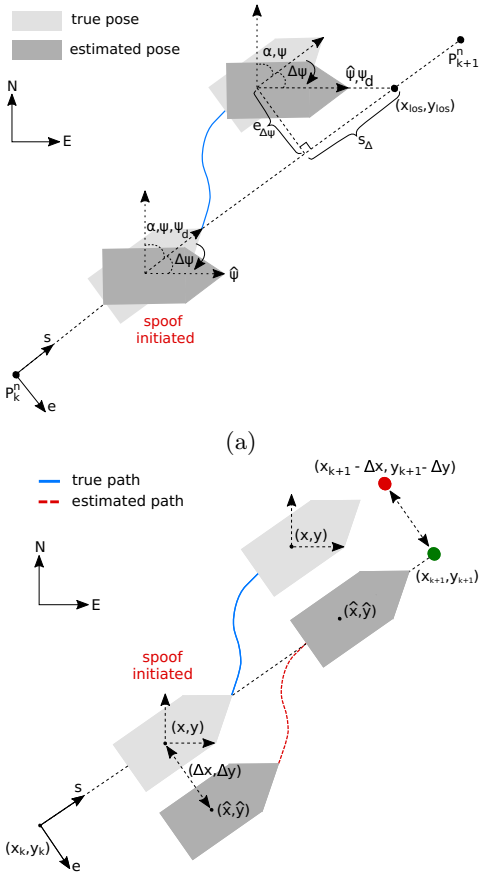


Figure 3.3 Illustrations of the expected behaviors of the USV when navigation signals are spoofed. External disturbances are neglected. (a) Expected behavior when the yaw of an underactuated USV is spoofed. (b) Expected behavior when the position of an underactuated USV is spoofed.

where external disturbances are neglected. The yaw error used by the heading controller is then given by

$$\begin{aligned}\tilde{\psi} &= \psi_d - \hat{\psi} \\ &= \psi_d - \psi - \Delta\psi.\end{aligned} \tag{3.5}$$

Consequently, the control system steers the yaw to $\psi = \psi_d - \Delta\psi$ to minimize (3.5). As such, the USV will pursue a path parallel to the desired path, with a cross-track error given by

$$e_{\Delta\psi} = -s_\Delta \tan \Delta\psi. \tag{3.6}$$

Hence, we see that adding an offset $\Delta\psi$ to the yaw results in a predictable change in the USV path. Similarly, if an adversary manages to spoof the position of the vehicle by an offset $(\Delta x,\ \Delta y)$, we have

$$(\hat{x},\ \hat{y}) = (x + \Delta x,\ y + \Delta y), \tag{3.7}$$

where external disturbances are neglected. Using (3.1), this translates to offsets in the path-fixed frame as

$$[\Delta s,\ \Delta e]^T = R_n^p(\alpha_k)[\Delta x,\ \Delta y]^T. \tag{3.8}$$

Consequently, assuming a lookahead-based LOS guidance system is used, the guidance system seeks to steer the vehicle towards the desired heading

$$\psi_d = -\text{atan2}(e + \Delta e, s_\Delta) \tag{3.9}$$

sending the vehicle to $(x_{k+1} - \Delta x,\ y_{k+1} - \Delta y)$. Illustrations of the expected behavior when the yaw or the position is spoofed are seen in Figs. 3.3a, and 3.3b, respectively.

3.3.3 Technical implementation

An interesting attack vector against distributed GNC systems communicating over IP is to redirect the traffic through a device that selectively changes the transmitted data to manipulate the vehicle. For example, the adversary can redirect the traffic by spoofing the Address Resolution Protocol (ARP). The purpose of the ARP is to associate an IP address to a link layer address, and an ARP spoof attack works by falsely associating the link layer address of the Man-in-The-Middle (MiTM) device with the IP address of the intended recipient. Consequently, the adversary can force all traffic to pass through the MiTM device. An illustration of the attack is shown in Fig. 3.4.

Injection attack

Assuming a distributed GNC architecture is used, we can connect a single-board computer to an insecure switch and use ARP spoof to redirect the traffic going from the navigation system to the guidance & control system. The computer then runs a script where the contents of the IP packets are analyzed. The IP packets

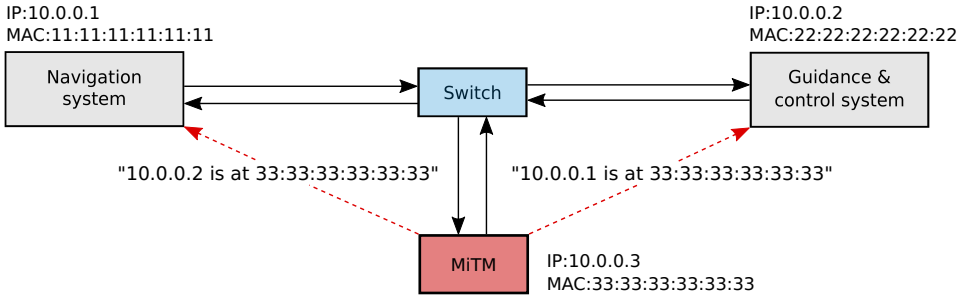


Figure 3.4 A MiTM device that redirects traffic by spoofing the ARP.

that do not contain the navigation parameter of interest are passed through to the intended recipient, while the IP packets containing the navigation parameter are manipulated. This is possible since the content and the structure of the unencrypted messages are available to the adversary. We use the Python packages NFQUEUE [127] and SCAPY [128] to intercept, inspect, manipulate, and re-transmit IP packets.

In this case study, the IMC protocol is used to transmit messages. An important observation is that the only integrity check on the IMC messages is a CRC-16 code computed using the generator polynomial $p(x) = x^{16} \oplus x^{15} \oplus x^2 \oplus 1$ with coefficients in the finite field GF(2). Therefore, we can change the message content, after which we forge and append a new, valid CRC-16 code to the message. In Python, CRC functions are readily available using the package CRCMOD. Pseudocode describing the MiTM injection attack can be seen in Algorithm 3, where navigation data is manipulated by adding an offset to the navigation parameter.

Algorithm 3 Man-in-the-Middle Injection Attack

θ: navigation parameter
Δ_θ: parameter offset
Message: (Header || Payload || Footer)
Execute ARP spoof
Initialize CRC-16
for each intercepted Message **do**
    if Message contains θ **then**
        $\theta \leftarrow$ ReadParameter(Payload)
        $\theta \leftarrow \theta + \Delta_\theta$
        Payload $\leftarrow$ WriteParameter(Payload, θ)
        Footer $\leftarrow$ CRC-16(Header || Payload)
        Message $\leftarrow$ (Header || Payload || Footer)
    end if
    Transmit(Message)
end for

Replay attack

While we can prevent injection attacks with MACs, solely using MACs does not prevent injection of previously transmitted, valid (message, tag)-pairs, and we are therefore vulnerable to *replay attacks*. In a replay attack, a set of authenticated messages can be recorded and later replayed. Since the messages are not changed, the authentication tag is still valid, and the receiver accepts the messages upon reception. Actively steering the vehicle using replayed messages is more challenging since the content of the messages is not necessarily known to the adversary. However, replay attacks can still disrupt the path of the USV. If the messages are encrypted, the data type of the message is more challenging to determine. However, it may still be possible by inspecting the metadata, for example, the size of the packets. Alternatively, all traffic can be logged and replayed. An example of a replay attack is shown in Algorithm 4, where messages are recorded over a pre-determined time interval and then immediately replayed.

Algorithm 4 Man-in-the-Middle Replay Attack

θ: navigation parameter
t: time since initialization
τ: duration of replay attack
Message: (Header || Payload || Footer)
Q: queue for messages
Execute ARP spoof
for each intercepted Message **do**
    if Message contains θ **and** t $<$ τ **then**
        Q.enqueue(Message)
    else if Message contains θ **then**
        Message $\leftarrow$ Q.dequeue()
    end if
    Transmit(Message)
end for

Securing the navigation data

To secure the navigation data against injection and replay attacks, we can use authenticated encryption with the addition of timestamps or sequence numbers. In our example, we add a fresh timestamp to the navigation data before both are encrypted. We then compute a MAC tag over the resulting ciphertext, the header of the message, and the IV. Upon reception, we recompute the MAC tag and decrypt the navigation data and the timestamp. If the recomputed and received tags match and the timestamp is fresh, the navigation data is accepted. An illustration of the signal flow with the proposed secure transmission and reception algorithms is shown in Fig. 3.5, and pseudocodes for the secure transmitter and receiver are found in Algorithms 5 and 6, respectively. We use the authenticated encryption algorithm AEGIS, a cryptographically strong authenticated encryption algorithm that has

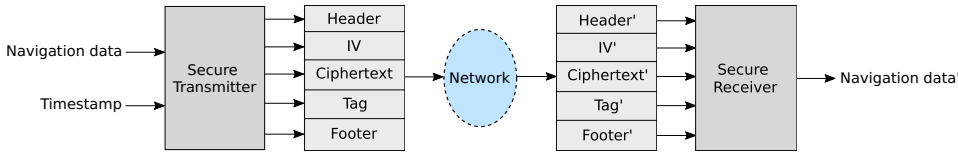


Figure 3.5 A flow chart of secured communication between the navigation system and the guidance & control system.

been shown to provide excellent performance in software with negligible time delays [74]. The AEGIS implementation used is publicly available and described by [79].

Algorithm 5 Secure Transmitter

K: Symmetric key; IV: Initialization Vector;
$AE_{K,IV}$: Authenticated encryption function parameterized by K and IV;
SecureMessage: (Header || Payload || Footer)
T: Timestamp

```
1:  Initialize CRC-16
2:  while true do
3:      Initialize AE_{K,IV}
4:      Instantiate SecureMessage
5:      NavigationData ← ReadNavigationData()
6:      T ← GetTime()
7:      (Header || IV || Ciphertext || Tag) ← AE_{K,IV}(Header || T || NavigationData)
8:      Footer ← CRC-16(Header || IV || Ciphertext || Tag)
9:      SecureMessage ← (Header || IV || Ciphertext || Tag || Footer)
10:     Transmit(SecureMessage)
11:     Update IV
12: end while
```

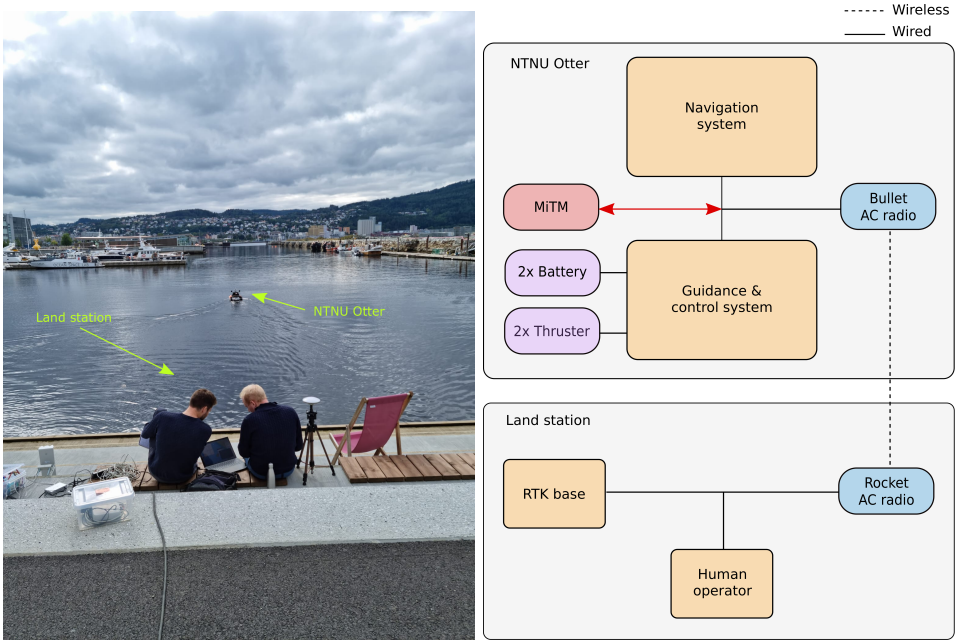
3.4 Experimental setup

The experimental setup consists of the Cyberotter USV and a land station. The Cyberotter uses a distributed GNC system in which the navigation and guidance & control system are two separate systems that communicate over ethernet. Furthermore, we assume that an adversary has gained access to the signal transmission onboard the Cyberotter between the navigation and guidance & control system. The land station consists of a real-time kinematic (RTK) base station that sends correction data to the navigation system and a remote laptop for the operator to upload missions or control the USV directly. The land station and the Cyberotter communicate using a point-to-point transparent ethernet bridge established over radio communication. Figure 3.6a shows an overview of the experimental scene, and Fig. 3.6b shows a schematic of the experimental setup.

Algorithm 6 Secure Receiver

K: Symmetric key; IV: Initialization Vector;
$AD_{K,IV}$: Authenticated decryption function parameterized by K and IV;
SecureMessage: (Header || Payload || Footer)
T: Timestamp

1: T = 0
2: **for** each received Message **do**
3: (Header' || IV' || Ciphertext' || Tag' || Footer') ← Read(Message)
4: Initialize $AD_{K,IV'}$
5: (T' || NavigationData' || Tag) ← $AD_{K,IV'}$(Header' || IV' || Ciphertext' || Tag')
6: **if** Tag == Tag' **and** T' > T **then**
7: T ← T'
8: Accept NavigationData'
9: **else**
10: Reject NavigationData'
11: **end if**
12: **end for**



(a) (b)

Figure 3.6 (a) An overview of the experimental scene showing the base station and the Cyberotter USV. (b) A high-level schematic of the land station and the USV.

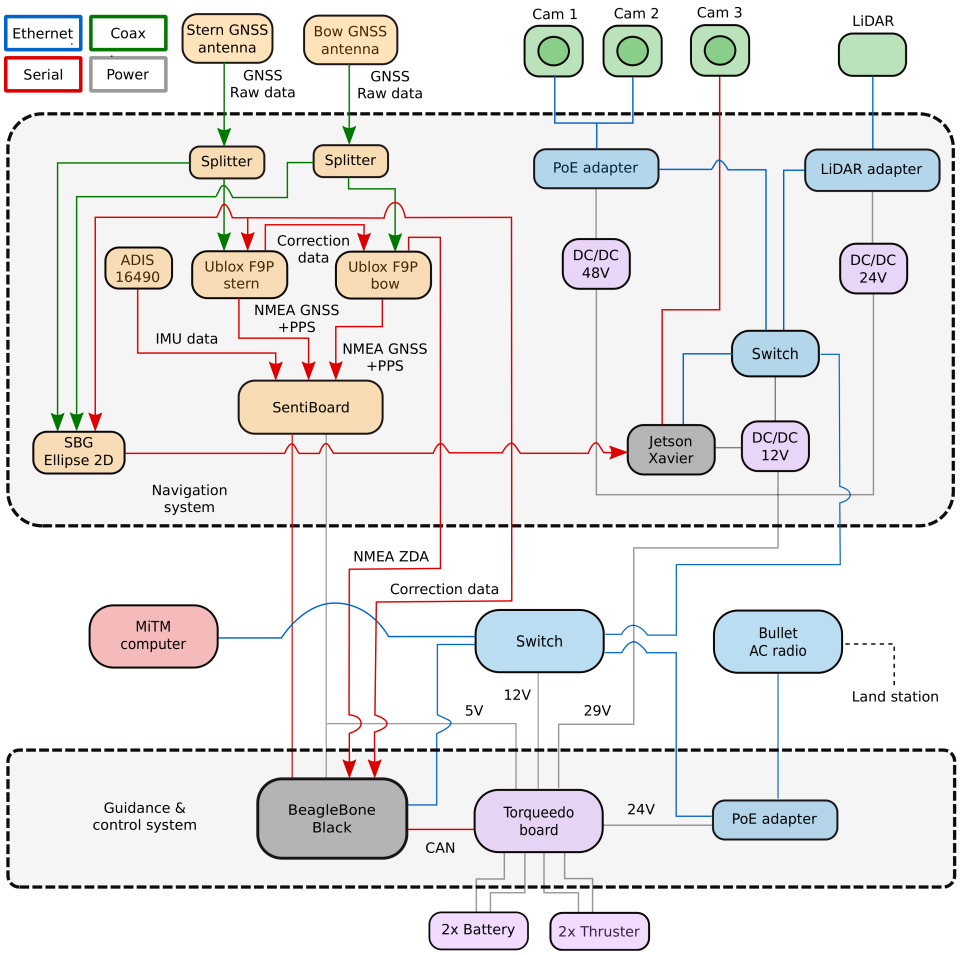


Figure 3.7 A hardware schematic of the navigation system and guidance & control system components of the Cyberotter USV.

3.4.1 The Cyberotter

The Cyberotter is underactuated with fixed starboard and port thrusters mounted at the stern. The software and hardware architectures were designed and built at the Department of Engineering Cybernetics, NTNU, while the body, thrusters, batteries, and the power interface board were purchased from Maritime Robotics AS. A schematic of the hardware onboard the Cyberotter is shown in Fig. 3.7. We use the LSTS software toolchain, consisting of Dynamic Unified Navigation Environment (DUNE), the IMC protocol, and the Neptus command and control software, to control and interact with the vehicle. DUNE is used for guidance, control, and navigation and to interface with hardware components, while the IMC protocol is used to transmit data between individual DUNE tasks. Finally, we use Neptus to interact with the vehicle by passing maneuvers to the guidance system

or remote controlling the USV from the land station.

Navigation system

The Cyberotter uses two independent navigation systems. The first navigation system consists of an ADIS 16490 IMU [129] and two U-blox F9P GNSS receivers [130] with synchronized data acquisition through a SentiBoard [131]. The first GNSS receiver is configured as a 'moving base' and receives raw GNSS data from an antenna mounted at the stern of the Cyberotter and correction data from the RTK base. The second GNSS receiver is configured as a 'rover' and receives raw GNSS data from an antenna mounted at the bow of the Cyberotter and correction data from the moving base. As such, the rover finds the yaw of the USV. The second navigation system consists of an SBG Ellipse 2D INS [132], which receives raw GNSS data from the stern and bow antennas and correction data from the base station. For our experiments, the navigation data from the SBG Ellipse 2D was used in feedback control, while the navigation data from the SentiBoard was used as ground truth measurements for comparison. Since the navigation systems receive corrections from the same base with centimeter precision, the navigation data produced are almost identical. As such, the effect of measurement noise is reduced to a minimum. The navigation system also contains vision-based sensors, that is, cameras and a LiDAR, that can be used for local navigation purposes. However, these sensors are not used for the experiments. A schematic of the complete navigation system can be seen in the upper part of Fig. 3.7.

Guidance system

The guidance system consists of a path planner and a LOS guidance law with integral action, that is, ILOS [133]. The guidance system receives a set of waypoints and desired speeds from the operator, and the estimated position, velocity, and yaw, from the navigation system. The path planner then produces the desired path using the waypoints and the desired speeds. Then, the ILOS guidance law computes the desired yaw based on the estimated state and the desired path. Using the following condition

$$\frac{\sqrt{(||R_n^p(p_{k+1}^n - p_k^n)||_2 - s)^2 + e^2}}{u} - C_t \leq 0, \tag{3.10}$$

the path planner determines whether a waypoint has been reached or not. Here, C_t is a positive constant. To avoid problems with integral windup resulting in large overshoots, the integral action of the ILOS guidance law is only used when the USV is located within a certain distance from the desired path [134]. In practice, we use a cross-track distance of 2.5 m to determine whether integral action is enabled or not.

Control system

The control system consists of a proportional-integral speed controller and a proportional heading controller. Based on the estimated state from the navigation

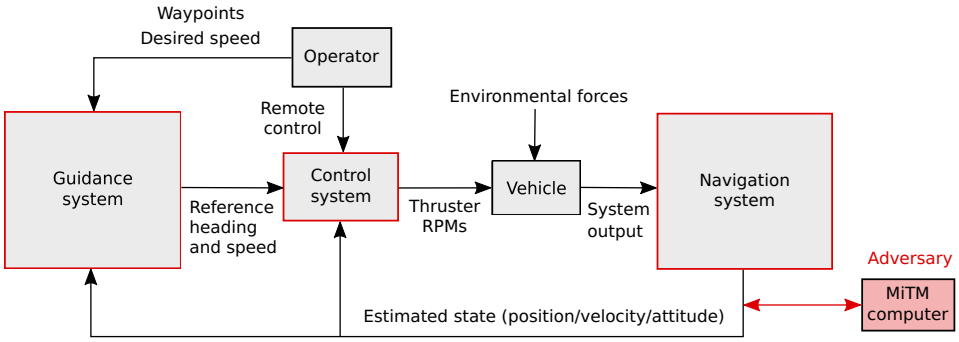


Figure 3.8 The closed-loop GNC system of the Cyberotter USV under attack by a MiTM adversary.

system and the desired speed and yaw from the guidance system, the control system produces desired revolutions per minute of the starboard and port thrusters. The speed controller contains logic that disables the controller if the difference between desired and estimated yaw exceeds 36° to reduce the cross-track error following sharp turns. The control system also permits remote operation, in which manual control signals can be transmitted from a PlayStation® 4 (PS4) controller connected to the remote control laptop. An illustration of the signal flow of the closed-loop system is shown in Fig. 3.8.

Synchronization

We use the Precision Time Protocol (PTP) to synchronize the hardware clocks onboard the Cyberotter. With PTP, the devices are synchronized with sub-microsecond precision using a master-slave setup [135]. We configure the Beaglebone Black computer [136] in the guidance & control system to be the master clock, and we configure the Jetson Xavier computer [137] in the navigation system to be the slave clock. The master clock derives the time from a GNSS receiver using the NMEA ZDA message, as shown in Fig. 3.7. Furthermore, we use a SentiBoard for data synchronization. The SentiBoard is synchronized with Coordinated Universal Time (UTC) using a Time of Validity (TOV) signal, often referred to as the pulse per second, from the GNSS receivers. The IMU also produces a TOV each cycle, after which the SentiBoard reads and timestamps the IMU data, in hardware, with its internal clock. With this setup, the data is synchronized to UTC with a root mean squared clock drift of 1.9 μs per second [138].

3.4.2 Land station

The land station consists of a remote control computer running the Neptus command and control and an RTK base station that transmits corrections to the navigation system. We used the remote control computer to create and upload missions to the guidance system or control the vehicle manually with a PS4 controller. The RTK base station consists of a GNSS antenna, an U-blox F9P GNSS receiver, and

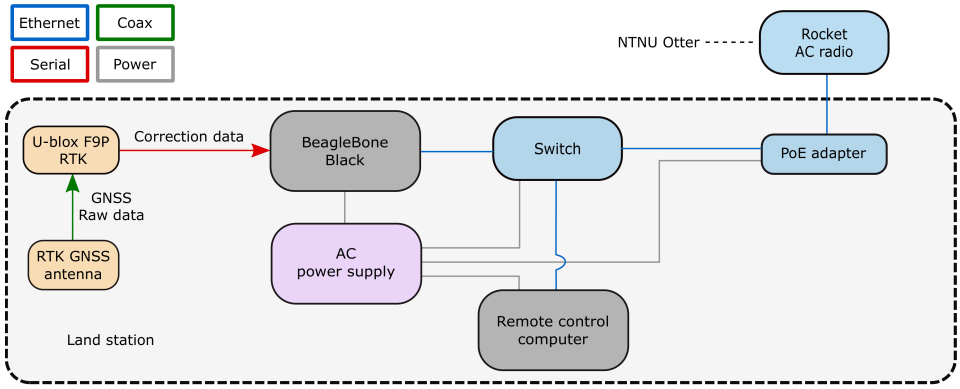


Figure 3.9 A hardware schematic of the land station components.

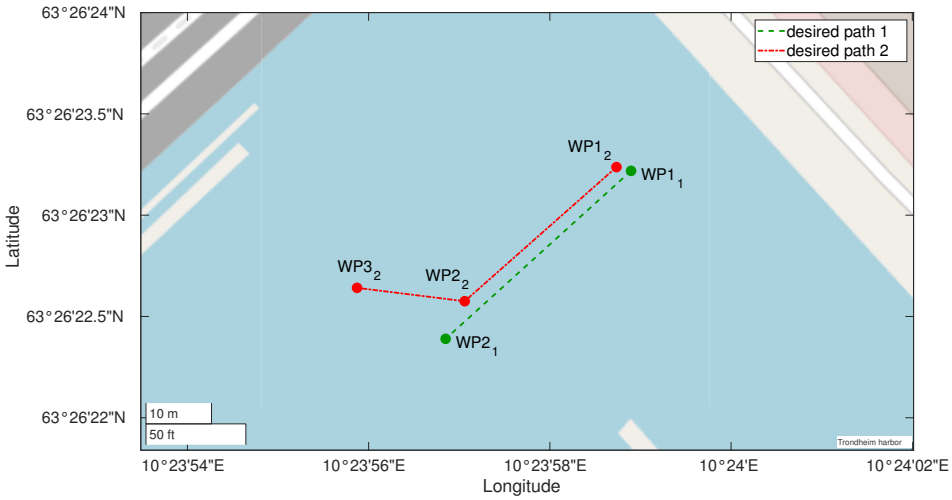


Figure 3.10 Predefined waypoints determine the desired paths of the vehicle used in the experiments.

a Beaglebone Black, as shown in Fig. 3.9. We configured the GNSS receiver to estimate the phase of the GNSS carrier wave over 17 hours before we conducted the experiments. This surveying procedure resulted in an absolute precision of 6 cm, negatively affected by a cruise ship that docked close to the GNSS antenna during the survey.

3.4.3 Experimental description

We perform five field experiments to demonstrate the vulnerability of the distributed GNC system onboard the USV in the harbor environment. The desired paths of the vehicle during the experiments are shown in Fig. 3.10. Experiments 1-4 are

conducted with desired path 1, where the desired speed between $WP1_1$ and $WP2_1$ is set to 0.5 m/s and 0.25 m/s in Experiments 1 and 2 and Experiments 3 and 4, respectively. Experiment 5 was conducted using desired path 2 with desired speed set to 0.5 m/s between the waypoints.

We manipulate the vehicle by adding fixed offsets to the yaw and latitude estimates in Experiments 1 and 2, respectively. In Experiment 1, we alter the heading by adding a fixed offset of 57.3°, and in Experiment 2, we change the latitude by adding a fixed offset of approximately 10 m. Since large offsets are easy to detect, we also implement attacks where the yaw and latitude are changed by incremental offsets, slowly dragging the vehicle off course. Consequently, we manipulate the vehicle by adding incremental offsets of 0.573° per second and 0.07 m per second to the yaw and the latitude in Experiments 3 and 4, respectively. The speed was lowered to 0.25 m/s for the incremental spoofing attacks to take effect over an extended period. In Experiments 1-4, we initiate the attacks when the vehicle is between $WP1_1$ and $WP2_1$. We proceed by performing a replay attack in Experiment 5, where a sequence of encrypted and authenticated messages containing heading information from the navigation system is recorded and replayed with a 30-second delay to manipulate the vehicle. The vehicle heading is recorded between $WP1_2$ and $WP2_2$ and replayed just before the planned course change at $WP2_2$. We use the second path in this experiment to see how the vehicle handles the planned course change while receiving delayed heading information.

We include three scenarios for each experiment. First, we execute a reference scenario to observe how well the vehicle follows the path while affected by environmental forces such as winds and currents. We then perform an attack scenario to show how the USV is affected by the attack. Finally, we execute a secured scenario to see how well the added countermeasures protect the vehicle against the attacks.

3.5 Experimental results

We present the results of the experiments by plotting the USV position during the attack scenario against the position of the vehicle in the reference scenario and the secured scenario. The manipulated parameter, that is, heading or position, is plotted against the true value of the parameter obtained by the redundant navigation system. When the heading is spoofed, we also plot the desired heading from the guidance system. At last, we show the effect of using the proposed secure transmitter and receiver, described in Algorithms 5 and 6.

3.5.1 Experiment 1: Fixed heading spoof

The results from Experiment 1 are shown in Fig. 3.11. Figure 3.11a shows how the vehicle deviates from the desired path, and Fig. 3.11b shows how the estimated heading changes after adding a fixed heading offset. When we secure the signal transmission with authenticated encryption, the spoofing attack is detected immediately, and all spoofed messages are dismissed. The control system uses the latest heading estimate available before the attack. As a result, the vehicle continues along the desired path with an oscillating heading, as shown in Fig. 3.11c.

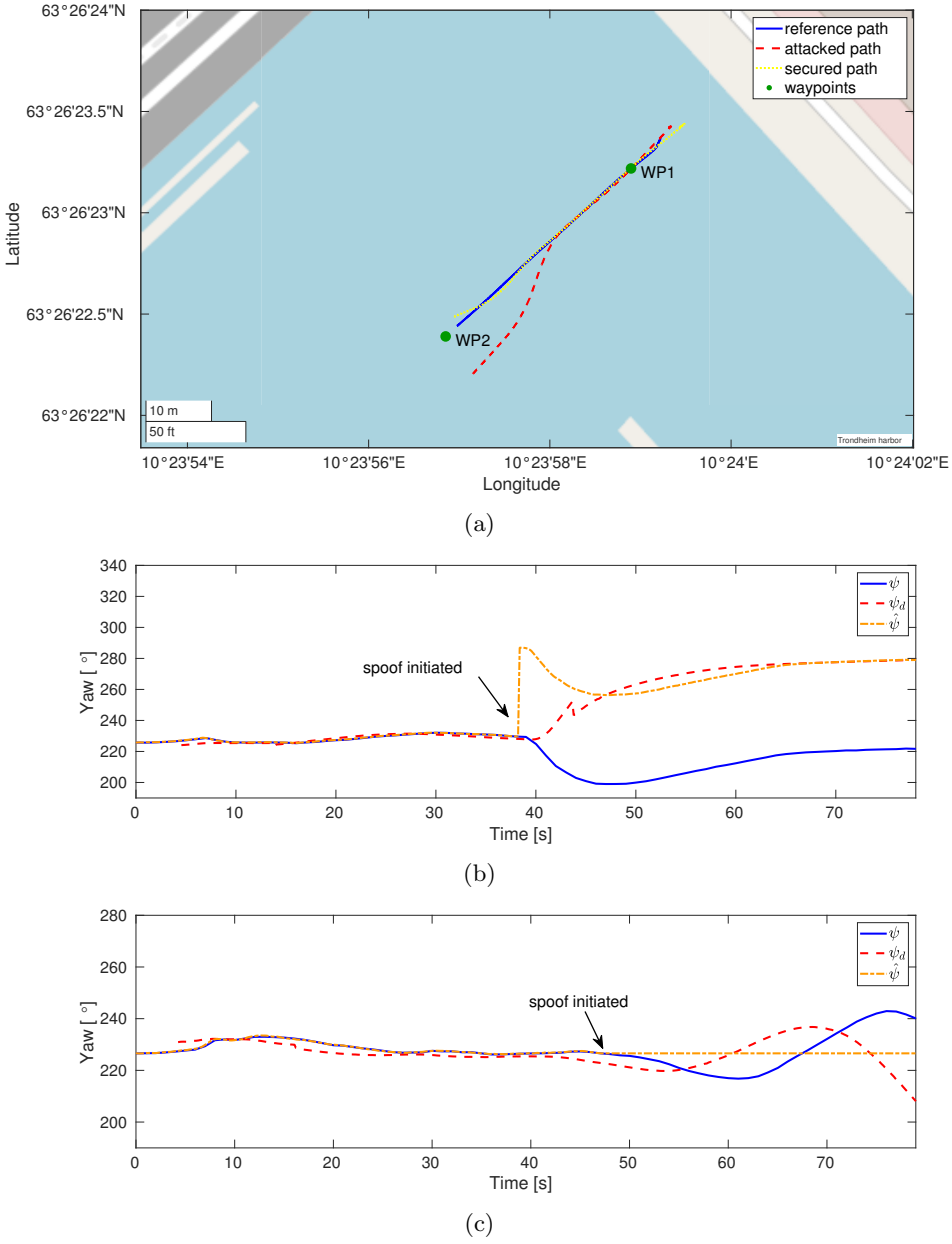


Figure 3.11 The results from Experiment 1. (a) The paths between the waypoints of the USV in the three scenarios. (b) True, desired, and estimated heading of the USV when we attack the insecure system with a fixed heading offset. (c) True, desired, and estimated heading of the USV when we attack the secured system with a fixed heading offset.

3.5.2 Experiment 2: Fixed latitude spoof

The results from Experiment 2 are shown in Fig. 3.12. Figure 3.12a shows how the fixed latitude spoof successfully puts the vehicle off course. In Fig. 3.12b, we plot the true and the estimated paths of the USV during the attack scenario, showing the sudden jump in the estimated position when we launch the attack. When we secure the signal transmission with authenticated encryption, the spoofing attack is detected, and all spoofed messages are dismissed. Without updated position estimates, the vehicle goes to an error state, and the mission is aborted.

3.5.3 Experiment 3: Incremental heading spoof

The results from Experiment 3 are shown in Fig. 3.13. Figure 3.13a shows the paths of the vehicle in the three scenarios. The effect of the incremental attack is not immediately visible on the path of the vehicle in the attack scenario. However, the vehicle veers off course in an attempt to correct its heading towards the end. The increasing deviation between the true and estimated heading of the vehicle is visible in Fig. 3.13b. When the signal transmission is secured with authenticated encryption, the spoofing attack is detected and all spoofed messages are dismissed. Similar to Experiment 1, the vessel continues along its desired path; however, the heading oscillations are more pronounced because the USV operates without an updated heading estimate for an extended time period, as can be seen in Fig. 3.13c.

3.5.4 Experiment 4: Incremental latitude spoof

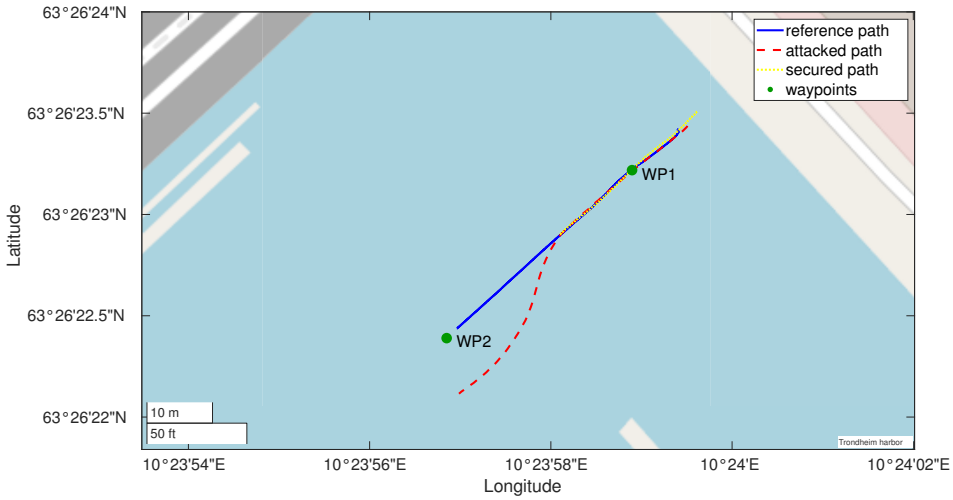
The results from Experiment 4 are shown in Fig. 3.14. Figure 3.14a shows the paths of the vehicle in the three scenarios. We successfully drag the USV off course in the attack scenario by adding an incremental offset to the latitude estimate. We show this in Fig. 3.14b, where we plot the true and the estimated path of the USV. When we secure the system using authenticated encryption, the spoofed messages are dismissed, and the vehicle enters an error state. Consequently, the mission is aborted.

3.5.5 Experiment 5: Replay attack

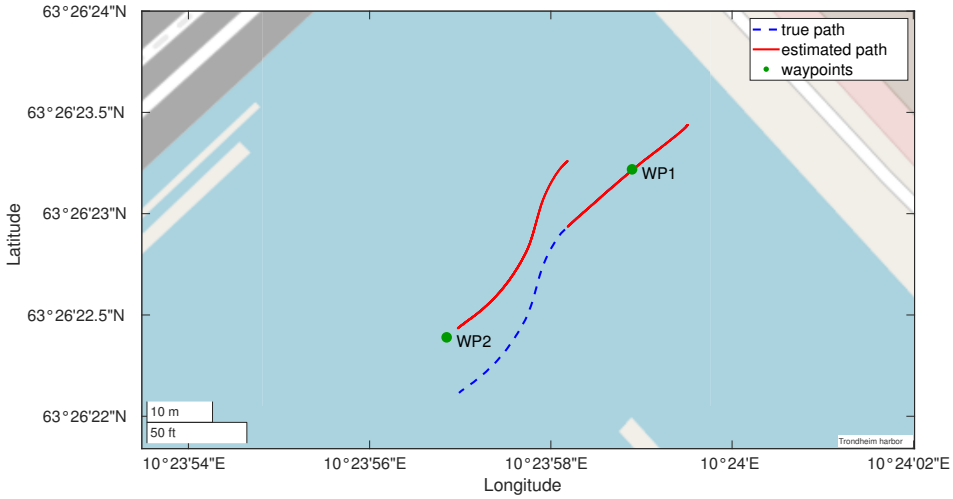
The results from Experiment 5 are shown in Fig. 3.15. Figure 3.15a shows the paths of the vehicle in the three scenarios. The replay attack is seen to cause a slightly delayed action compared to the reference path. Furthermore, Fig. 3.15b shows that the replay attack successfully changes the estimated heading immediately before the USV reaches $WP2_2$. When we secure the system by adding authenticated timestamps, the replayed messages are identified and discarded. Consequently, the vehicle enters an error state, and the mission is aborted.

3.5.6 Discussion of results

The experiments demonstrated that ARP spoof is an effective attack vector against distributed GNC systems communicating over a local network. Furthermore, they showed that messages transmitted using protocols merely relying on conventional

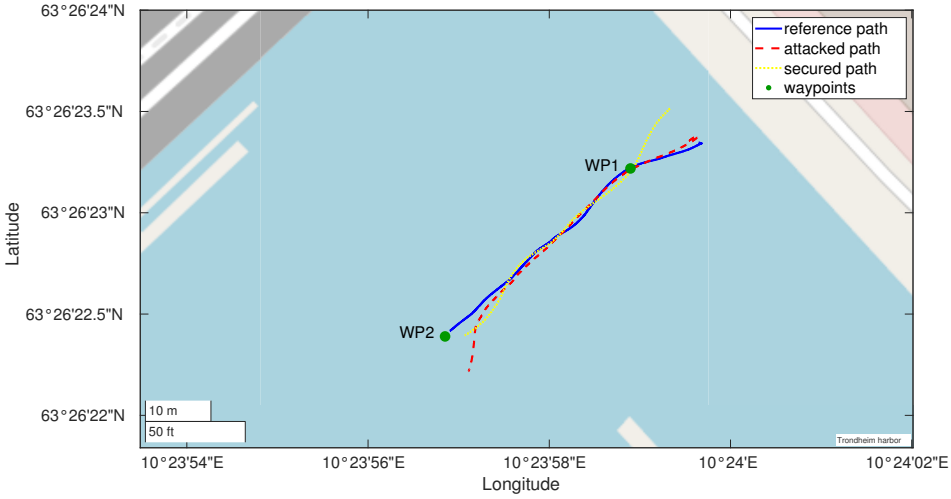


(a)

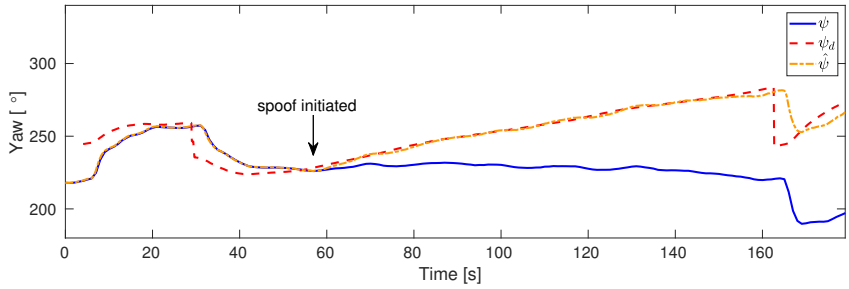


(b)

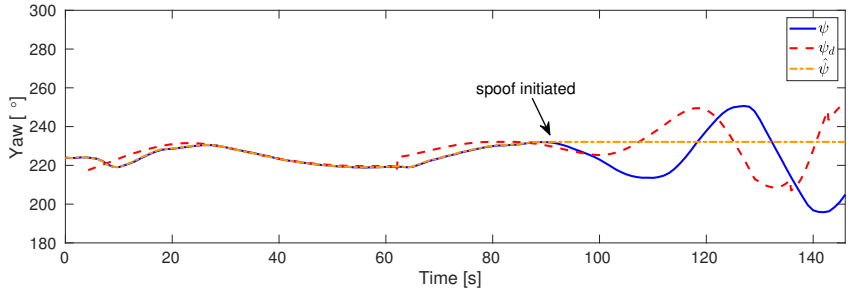
Figure 3.12 The results from Experiment 2. (a) The paths between the waypoints of the USV in the three scenarios. (b) True and estimated path of the USV when we attack the insecure system with a fixed latitude offset.



(a)

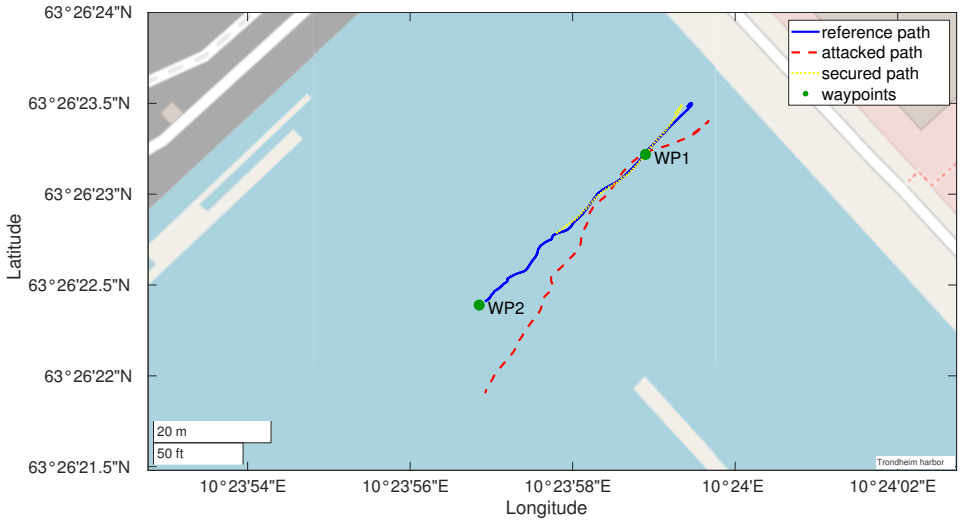


(b)

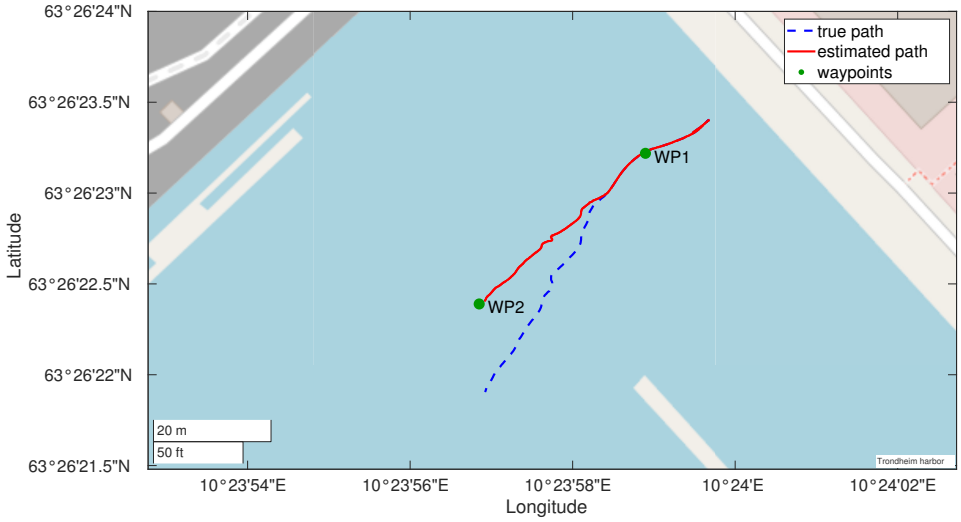


(c)

Figure 3.13 The results from Experiment 3. (a) The paths between the waypoints of the USV in the three scenarios. (b) True, desired, and estimated heading of the USV when we attack the insecure system with an incremental heading offset. (c) True, desired, and estimated heading of the USV when we attack the secured system with an incremental heading offset.

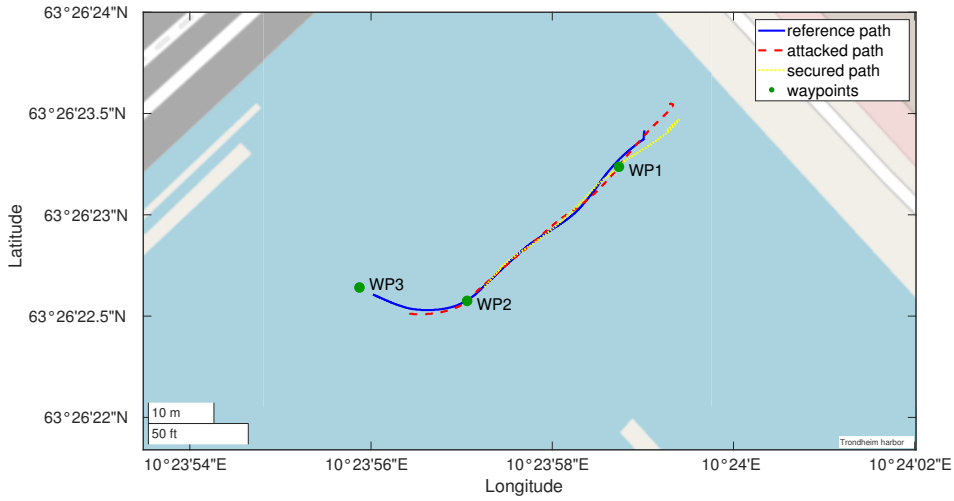


(a)

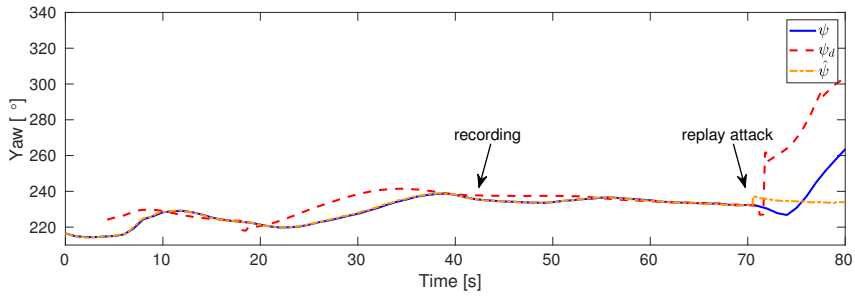


(b)

Figure 3.14 The results from Experiment 4. (a) The paths between the waypoints of the USV in the three scenarios. (b) True and estimated path of the USV when we attack the insecure system with an incremental latitude offset.



(a)



(b)

Figure 3.15 The results from Experiment 5. (a) The paths between the waypoints of the USV in the three scenarios. (b) True, desired, and estimated heading of the USV when we attack the insecure system with a replay attack.

fault checks to detect invalid messages are vulnerable to eavesdropping and injection attacks. As expected, manipulation of heading and position estimates caused a predictable change in the path of the underactuated USV. Notice that, while we here considered attacks where the spoofed values were computed by adding offsets, more advanced methods of selecting the spoofed values can be used without fundamentally changing the attack. Furthermore, we demonstrated that authentication is insufficient to prevent the successful injection of recorded messages through a replay attack. Finally, we showed that authenticated encryption with timestamps effectively prevents the hijacking of the USV.

When the attacks caused the USV to deviate from the desired path, the integral effect of the ILOS guidance was switched off when the cross-track distance exceeded the 2.5 m threshold. We see this in Figs. 3.11b and 3.13b, where we observe sudden jumps in the desired heading when the heading was spoofed. The effect is especially

pronounced in the incremental heading spoof. We believe this is because the integral action had been enabled over an extended period, and the weather conditions were worsening with strong wind gusts and currents, causing a varying crab angle. The varying crab angle also strongly influenced the incremental latitude spoof, with the USV oscillating around the desired path, as seen in Fig. 3.14a. It is clear that the assumption that the crab angle would be slowly varying, used by the ILOS guidance law, was not satisfied during these experiments.

To secure the USV against the hijacking attempts demonstrated in Experiments 1-4, Algorithms 5 and 6 were used by the navigation system and guidance and control systems, respectively. When an attack was detected, we used two separate failure modes for position spoofs and heading spoofs. When a position spoof was detected, the USV went to a failure mode and halted all actions when no recent valid position estimates were available. In contrast, heading spoofs were handled by using the most recent, valid heading estimate available. Notably, the latter resulted in oscillating behavior of the USV, as seen in Figs. 3.11a and 3.11c, and Figs. 3.13a and 3.13c. Consequently, we found that if a heading spoof is detected and no new heading estimates are available, the USV should, instead, go to a failure mode to prevent erratic behavior. We believe that more extensive safety analysis, and the development of relevant failure modes, are important steps towards safe autonomous vehicles.

As shown from the desired and the true heading in Fig. 3.15b, the replay attack resulted in delayed action of the heading controller. This delay resulted in a slight change of the path. From Fig. 3.15a, we also observed that the mission was fulfilled approximately 8 m before $WP3_2$. Because of the planned course change in $WP2_2$, the difference between the desired and the estimated heading exceeded a 36° threshold, at which point the speed controller was switched off. This resulted in a significant increase in surge speed u. Consequently, because of the increase in surge speed, the inequality (3.10) was satisfied early, and the path planner prematurely announced that the mission had been completed. In the reference path, the speed controller largely remained on, and the USV got much closer to $WP3_2$ before the path planner announced that the mission had been completed. Unfortunately, the distance between $WP2_2$ and $WP3_2$ was not sufficiently large to fully capture the consequence of the attack. Nevertheless, the attack successfully changed the estimated heading of the USV. When the communication was secured using Algorithms 5 and 6, the replay attack was immediately detected. When messages with old timestamps were detected, and no fresh heading estimates were available, the USV aborted the mission and went into an error state instead of continuing along the desired path.

It is clear that when Algorithms 5 and 6 were used, the attacks still managed to take the USV out of service. However, when an adversary gains access to the transmission lines of the GNC system, Denial of Service (DoS) attacks are trivial to execute. Additionally, the proposed algorithms do not prevent delay attacks where the MiTM device merely delays messages instead of replaying them. However, actively steering the vehicle through such an attack with encrypted messages is highly unlikely since the device has no means of knowing the contents of the delayed messages. Consequently, we classify this as a DoS attack. Possible methods to detect such attacks range from comparing the interval between received messages

to an expected value and comparing timestamps on received messages to the local clock. Importantly, keeping the USV in service should not be the goal. Instead, the important takeaway is that spoofed and replayed messages are detected and discarded such that the vehicle cannot actively be steered, that is, hijacked, by the adversary.

3.6 Chapter summary

With recent advances in ICT paving the way for increased use of unmanned and autonomous vehicles, implementing secure GNC systems is crucial to ensure safe and reliable operation. Successful cyber-attacks, for example, as part of a terrorist attack, may cause fatal human and financial consequences and devastate trust by authorities, investors, and the general public. Previous studies have highlighted potential vulnerabilities in autonomous vehicles through surveys and high-level studies. Among the few studies demonstrating attacks against intra-vehicular communication, experimental verification has been limited to conventional vehicles and controlled laboratory environments. Hence, there is a gap between theory and practice in cybersecurity for autonomous vehicles. Furthermore, studies presenting countermeasures usually resort to anomaly-based IDSs. However, anomaly-based IDSs suffer from high false-positive rates, and because of the base rate fallacy, these systems are not appropriate for practical applications.

In this chapter, we have addressed these problems and verified and analyzed the effects of the proposed attacks and countermeasures through field experiments. Firstly, we have demonstrated how injection attacks can actively take control of an underactuated USV, thus bridging the gap between theory and practice. Secondly, we have shown how cryptographic methods effectively prevent attacks against intra-vehicular communication. Consequently, we recommend that developers actively secure intra-vehicular communication in GNC systems by using the proposed secure transmitter and receiver algorithms combining authenticated encryption, for example, the AEGIS framework, with additional plaintext redundancy, such as timestamps or sequence numbers, to prevent eavesdropping, injection, and replay attacks.

Part II

Applications of Homomorphic Encryption

Chapter 4

Development and Experimental Validation of an Encrypted Control System for an Unmanned Surface Vehicle

This chapter is based on the publication

[76] P. Solnør, S. Petrovic, and T. I. Fossen, "Development and experimental validation of an encrypted control system for an unmanned surface vehicle," *Robotics and Autonomous Systems*, pp. 1–17, 2023. (Submitted)

4.1 Introduction

Developments in communication and cloud computing technologies are paving the way for distributed computing services. Today, developers and businesses can upload and host software on public cloud infrastructures, and end-users can access the software via the internet. This concept is frequently referred to as *software as a service* [32]. In fact, by offloading and centralizing computational efforts, cloud computing is also seen as a promising technology for industrial applications such as process control and autonomous vehicles [139]. This motivates the development of cloud-enabled feedback control systems where part of the feedback loop can be hosted on cloud infrastructure [140, 141], and the use of such control systems for motion planning and control of robots is called *cloud robotics* [34, 142, 35].

Unfortunately, by introducing communication links and remote computation, cloud-based control systems become more exposed to cyberattacks such as eavesdropping, data injection, and replay attacks than conventional control systems. Adversaries can use data injection and replay attacks to manipulate physical processes and, for example, destabilize a chemical processing plant or hijack an autonomous vehicle. Eavesdropping attacks are a more subtle risk. They can be used, for example, for industrial espionage or as reconnaissance before more advanced attacks. We refer to feedback control systems that close the loop over a network as

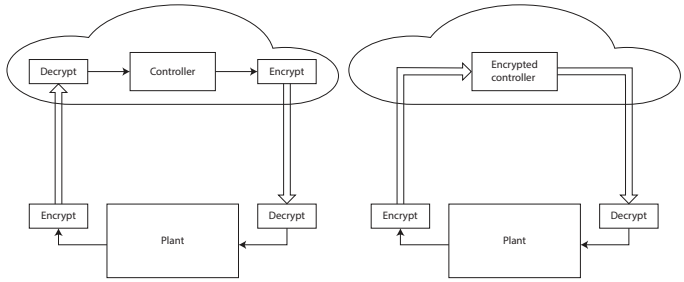


Figure 4.1 Encrypted vs Conventional Control. (a) A cloud-based feedback control loop secured with conventional cryptography. (b) A cloud-based feedback control loop with an encrypted controller.

NCSs, and the cybersecurity of such systems has gained much attention over the last decade [7, 143, 56, 92, 75]. Since the infrastructure on which the system modules run are usually trusted entities, it often suffices to incur secure data transmission between the modules with conventional cryptographic tools. However, hosting control systems on public cloud infrastructures poses new, unique challenges.

A key benefit of hosting software on public cloud infrastructure is that it allows access to existing computational resources of third parties, which is usually considerably cheaper than establishing dedicated infrastructures with similar capabilities. Typically, the cloud decrypts the data before subjecting it to mathematical operations, as shown in Fig. 4.1a, giving the third party access to the data stored on and passing through the cloud. However, if the data is considered confidential, this poses an unacceptable risk for information leaks. As a result, the development of secure cloud computing methods that preserve the confidentiality of the data involved while enabling mathematical operations is a subject undergoing intense study by cryptologists. One such method of ensuring privacy is to process the encrypted data directly without decrypting it on the cloud infrastructure, a technique made possible by homomorphic encryption.

In terms of feedback control, homomorphic encryption enables the development of encrypted control algorithms that can be hosted on cloud infrastructure without revealing state information or control parameters to third parties, as shown in Fig. 4.1b. For cloud robotics, these encrypted control systems are appealing since they adopt the benefits of cloud computing without revealing information, such as the robot's position, to the cloud. To this end, we seek to contribute to this development by designing and implementing an encrypted feedback control system for a USV and then demonstrating its efficiency through field experiments.

4.1.1 Related works

Pang et al. [31] first proposed conventional symmetric encryption to secure the signal transmission in networked control systems. It has since been demonstrated that the latency induced by state-of-the-art symmetric cryptography is very low

and, therefore, unlikely to pose a problem in feedback control, even when processing large quantities of data such as vision-based signals in real-time [74].

Kogiso and Fujita [56] were the first to propose the use of homomorphic encryption to secure networked control systems. Specifically, they used the multiplicatively homomorphic cryptosystems RSA[1] [50] and Elgamal [51] to build encrypted control systems. Control systems built using multiplicatively homomorphic cryptosystems can compute encrypted control inputs based on encrypted state estimates and encrypted control parameters. However, the use of multiplicatively homomorphic cryptosystems raises some problems. First, since only homomorphic multiplications can be performed, the encrypted controller can only transmit partial products back to the plant, which has to decrypt each partial product and perform summation in plaintext space. This necessarily increases the number of decryption operations and the bandwidth required. Second, constructing dynamic controllers with integral effect is impractical since the summation must be performed at the plant rather than in the encrypted controller. Hence, the round trip communication delay becomes part of the computational latency, severely limiting the controller frequency. Nevertheless, the use of the Elgamal cryptosystem to build encrypted controllers has continued because the former property is attractive in applications where both control parameters and the input to the controller are considered secret [144, 145, 146].

Farokhi et al. [58] were the first to suggest encrypted control systems designed using the additively homomorphic Paillier cryptosystem [52]. By design, additively homomorphic cryptosystems enable homomorphic multiplications with plaintext constants. As a result, additively homomorphic cryptosystems can be used to design *semi-encrypted* control systems with plaintext control parameters acting on encrypted state estimates, or vice versa, with plaintext state estimates and encrypted control parameters. The Paillier cryptosystem has been used to implement partially encrypted MPC [61, 59, 60] and encrypted cooperative control [63, 62]. However, because homomorphic multiplications between ciphertexts are not possible, additively homomorphic cryptosystems cannot be used to implement fully encrypted controllers.

Barbosa et al. [147] later extended additively homomorphic methods to allow a single homomorphic multiplication between ciphertexts through a concept called labeled homomorphic encryption, thus allowing the evaluation of second-degree multivariate polynomials. Contrary to the tradeoff between privacy and communication load used by multiplicatively homomorphic encryption methods, labeled homomorphic encryption allows homomorphic multiplications of ciphertexts at the cost of revealing which operations the controller performs on the plaintext data. Alexandru and Pappas [65] further extended the scheme to allow more than a single ciphertext multiplication and designed and implemented an encrypted Linear Quadratic Gaussian, with the Paillier cryptosystem serving as the additively homomorphic cryptosystem. However, the solution is restricted to the time-invariant solution since the matrices cannot be updated in ciphertext space. A related concept, called

[1] The multiplicatively homomorphic property only holds for the textbook version of RSA, which is rarely used since similar plaintexts are mapped to similar ciphertext, that is, is it not *semantically secure*.

linear homomorphic authenticated encryption, was proposed and demonstrated in a real-time encrypted control system for an unmanned aerial vehicle by Cheon et al. [73]. The proposed scheme uses labeled programs, like labeled homomorphic encryption, but only supports the evaluation of linear operations, that is, the scheme does not allow multiplication of ciphertexts. However, the proposed method is unique in that the operations performed by the controller are authenticated.

Other approaches used to realize encrypted control systems include the use of secret sharing and multi-party computation techniques [148], polynomial control using two-party computation [149], and fully homomorphic encryption [150]. However, most studies have focused on obtaining and proving theoretical results related to the stability of these encrypted control systems, for example, under various quantization schemes. As emphasized by Schulze Darup et al. [57], there is a lack of implementation studies that consider practical aspects of encrypted control systems. In addition to the demonstration by Cheon et al. [73], semi-encrypted feedback control systems were implemented by Farokhi et al. [72] and Tran et al. [71], whom both used the additively homomorphic Paillier cryptosystem to implement semi-encrypted controllers used to control a two-wheeled robot and an inverted pendulum, respectively, both in controlled laboratory environments. Finally, Teranishi et al. [151] presented the only study where a fully encrypted control system was designed using multiplicatively homomorphic encryption and evaluated in an industrial setting.

4.1.2 Main contributions

The main objective of this study is to design and implement a fully encrypted control system for a USV. We use labeled homomorphic encryption to implement an encrypted surge speed and yaw controller for the USV and then validate the implementation through field experiments in an uncontrolled environment, where the feedback control loop is closed over a mobile broadband connection. The results indicate that the proposed method produces significantly lower computational latencies than fully encrypted control systems designed using the multiplicatively homomorphic Elgamal cryptosystem. Moreover, we discuss practical considerations and limitations applicable to encrypted control systems in general. As a result, this is the first extensive study where an encrypted control system is implemented and verified in the field.

4.1.3 Outline

The rest of the chapter is structured as follows. We begin by introducing our notation and some necessary concepts from algebra and cryptography that we use in this chapter in Section 4.2. We then show how to translate control laws to encrypted form with labeled homomorphic encryption in Section 4.4. In Section 4.5, we design and implement an encrypted surge speed and yaw controller for a USV. We validate the encrypted controller through an extensive field experiment in Section 4.6 and discuss the obtained results and practical considerations in Section 4.7. Finally, Section 4.8 concludes the chapter.

4.2 Notation and elements from cryptography

We let $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{Z}^+$ denote the set of real numbers, integers, and non-negative integers, respectively. For $a, b \in \mathbb{Z}$, the operation $a \mid b$ reads 'a divides b', and $gcd(a, b)$ denotes *the greatest common divisor* of a and b. If $gcd(a, b) = 1$, we say that a is *co-prime* to b. If $a, b \in \mathbb{Z}$, we say that a is *congruent* to b modulo n if $n \mid (a - b)$, and we write this $a \equiv b \pmod{n}$. For any non-zero $n \in \mathbb{Z}^+$ and $a \in \mathbb{Z}$, we let a mod n represent the smallest integer in the set $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ congruent to a modulo n. Moreover, we let a mods p denote the *absolute smallest modulo* p, that is, $a \in \{\frac{-p}{2} ..., 0, ..., \frac{p-1}{2}\}$. For an element $a \in G$, we let a^{-1} denote the *multiplicative inverse* if it exists. When we use the term *cryptosystem*, we refer to a full specification of a cryptographic primitive meant to provide confidentiality, that is, complete information about the keys, the plaintext space, the ciphertext space, the encryption algorithm, and the decryption algorithm.

Let $G = (X, \star)$, $H = (Y, *)$ be two algebraic groups, and consider a map $\phi : X \mapsto Y$. We define a *homomorphism* as, for example, defined in [152, p. 533]:

Definition 1 (Group Homomorphism)**.** *A homomorphism from a group $G = (X, \star)$ to a group $H = (Y, *)$ is a mapping $\phi : X \mapsto Y$ such that $\phi(a \star a') = \phi(a) * \phi(a')$ $\forall\, a, a' \in X$.*

For some cryptosystems, the *encryption* operation is a homomorphism between the plaintext space and the ciphertext space. If the group homomorphism holds for $'+'$, we call these cryptosystems *additively homomorphic*. On the other hand, if the group homomorphism holds for $'\cdot'$, we call these cryptosystems *multiplicatively homomorphic*. Both additively and multiplicatively homomorphic cryptosystems are commonly referred to as *partially* homomorphic.

4.2.1 Quadratic residues and Legendre and Jacobi symbols

Given an integer $n \geq 2$ and $a \in \mathbb{Z}_n^\times$, we say that a is a *quadratic residue* modulo n if there exists an $x \in \mathbb{Z}_n^\times$ such that $x^2 \equiv a \pmod{n}$. If no such solution exists, we call a a *quadratic non-residue*. We denote the set of quadratic residues modulo n by Q_n and the set of quadratic non-residues modulo n by $\bar{Q}_n$. For any integer a, Euler's Criterion states that if p is an odd prime, a is a quadratic residue modulo p if and only if $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$. This defines the *Legendre symbol*

$$\left(\frac{a}{p}\right) := \begin{cases} 0, & \text{if } p|a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \bar{Q}_p. \end{cases} \tag{4.1}$$

The concept of quadratic residues can be generalized to *nth power residues*. Given integers $n, N \geq 2$ and $a \in \mathbb{Z}_N^\times$, we call a an nth power residue modulo N if there exists an $x \in \mathbb{Z}_N^\times$ such that $x^n \equiv a \pmod{N}$ and an nth power non-residue if no such x exists. Similar to Euler's Criterion, it holds that a is an nth power residue if and only if $a^{\frac{p-1}{gcd(n,p-1)}} = 1$, and the symbol is defined as $\left(\frac{a}{p}\right)_n := a^{\frac{p-1}{n}} \text{ mods } p$.

Given an odd integer $N \geq 3$ with prime factorization $N = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$, we define the *Jacobi symbol*

$$\left(\frac{a}{N}\right) := \left(\frac{a}{p_1}\right)^{e_1} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{e_k}, \tag{4.2}$$

and denote the multiplicative group of elements whose Jacobi symbol is 1 as J_N. We may now state the *quadratic residuosity problem* as defined in [153, p. 99]:

Definition 2 (Quadratic Residuosity Problem)**.** *Given an odd composite integer N and $a \in J_N$, decide whether or not $a \in Q_N$.*

Notably, determining if $a \in J_N$ can be done efficiently. However, no such efficient solution is known for the quadratic residuosity problem if the factorization of N is unknown. This gives rise to the *quadratic residuosity assumption*:

Definition 3 (Quadratic Residuosity Assumption)**.** *For a random element $a \in J_N$ it is hard to determine if $a \in Q_N$ if the factorization of N is unknown.*

The quadratic residuosity assumption is the foundation on which the security of the Goldwasser-Micali cryptosystem [154], the first probabilistic public-key cryptosystem, is based. We say that the Goldwasser-Micali cryptosystem is *semantically secure*, as defined in [153, p. 306]:

Definition 4 (Semantic security)**.** *A public-key encryption scheme is said to be semantically secure if, for all probability distributions over the plaintext space, whatever a passive adversary can compute in expected polynomial time about the plaintext given the ciphertext, it can also compute in expected polynomial time without the ciphertext.*

4.2.2 The Joye-Libert cryptosystem

The Joye-Libert cryptosystem [155] is an asymmetric cryptosystem that generalizes the cryptosystem by Goldwasser and Micali [154] and is based on the quadratic residuosity assumption. The cryptosystem inherits the semantic security of the Goldwasser-Micali cryptosystem and consists of a tuple $\left(\hat{\text{KeyGen}}, \hat{\text{Enc}}, \hat{\text{Dec}}\right)$, defined as in [156][2]:

- $\hat{\text{KeyGen}}(1^\lambda)$: Let k denote the size of each plaintext in bits. $\hat{\text{KeyGen}}$ randomly generates two primes p and q of approximately the same size such that $p \equiv 1 \pmod{2^k}$, and sets $N = pq$. It also picks $y \in J_N \setminus Q_N$. The public key is then $pk = \{N, y, k\}$, and the private key is $sk' = \{p\}$. The plaintext space is given by $\mathcal{M} = \mathbb{Z}_{2^k}$ and the ciphertext space is given by $\hat{\mathcal{C}} = \mathbb{Z}_N^\times$.
- $\hat{\text{Enc}}(pk, m)$: To encrypt a plaintext $m \in \mathcal{M}$, pick a random $x \in \hat{\mathcal{C}}$ and return $c = y^m x^{2^k} \bmod N \in \hat{\mathcal{C}}$

[2] Contains minor corrections from the original paper by [155].

- $\hat{\text{DEC}}(sk, c)$: To decrypt a ciphertext $c \in \hat{\mathcal{C}}$, the algorithm computes $z = (\frac{c}{p})_{2^k}$ and then finds $m \in \mathcal{M}$ such that the relation

$$z = \left[\left(\frac{y}{p} \right)_{2^k} \right]^m mods\ p$$

holds. In practice, Algorithm 7 is used as shown by Benhamouda et al. [156].

Algorithm 7 Joye-Libert decryption algorithm

Parameters
- sk Joye-Libert private key

Input
- c Joye-Libert ciphertext
- y Joye-Libert public-key element
- k Joye-Libert plaintext size in bits

Output
- m Plaintext of the form $(m_{k-1}, ...m_0)_2$

function $\hat{\text{DEC}}$(sk, c)
  $m \leftarrow 0$; $B \leftarrow 1$; $D \leftarrow y^{\frac{-(p-1)}{2^k}}$
  $C \leftarrow c^{\frac{p-1}{2^k}} \bmod p$
  for $j = 1$ to $k - 1$ **do**
    $z \leftarrow C^{2^{k-j}} \bmod p$
    if $z \neq 1$ **then**
      $m \leftarrow m + B$
      $C \leftarrow C \cdot D \bmod p$
    end if
    $B \leftarrow 2B$
    $D \leftarrow D^2 \bmod p$
  end for
  if $C \neq 1$ **then**
    $m \leftarrow m + B$
  end if
  return m
end function

Of course, for any $m \in \mathcal{M}$, it holds that

$$\hat{\text{DEC}}(sk, \hat{\text{ENC}}(pk, m)) = m.$$

Additionally, given $m_1, m_2 \in \mathcal{M}$ such that $m_1 + m_2 \in \mathcal{M}$, it holds that

$$\hat{\text{DEC}}(sk,\ \hat{\text{ENC}}(pk, m_1) \cdot \hat{\text{ENC}}(pk, m_2) \bmod N) = m_1 + m_2,$$

that is, the Joye-Libert encryption operation is a homomorphism from $(\mathcal{M}, +)$ to $(\hat{\mathcal{C}}, \cdot)$, and we say that the Joye-Libert cryptosystem is additively homomorphic. Since $\hat{\mathcal{C}}$ is a multiplicative group, Joye-Libert also permits homomorphic subtractions

by multiplying with the multiplicative inverse. Moreover, given $m, k \in \mathcal{M}$ such that $km \in \mathcal{M}$, it holds that

$$\hat{\text{DEC}}(sk, \hat{\text{ENC}}(m, N)^k \bmod N) = km,$$

that is, we may perform homomorphic multiplication with plaintext constants. We will denote homomorphic addition with $\boxplus$, homomorphic subtraction with $\boxminus$, and homomorphic multiplication with plaintext constants with $\odot$, while $+$, $-$, and $\cdot$ denote conventional addition, subtraction, and multiplication in $\mathcal{M}$ or $\hat{\mathcal{C}}$. Ciphertexts in $\hat{\mathcal{C}}$ will henceforth be denoted by a lowercase c.

4.3 Labeled homomorphic encryption

Barbosa et al. [147] showed how to extend additively homomorphic cryptosystems to permit a multiplication between two ciphertexts, allowing the evaluation of second-degree multivariate polynomials. This is achieved by assigning unique labels to each variable and giving the decryption operation access to the polynomial evaluated and the labels associated with each input variable. The resulting cryptosystem is called *labeled homomorphic encryption*, and the pairing of a polynomial and its associated labels is called a *labeled program*:

Definition 5 (Labeled program)**.** *A labeled program $\mathcal{P}$ is a tuple $(f, \tau_1, \ldots, \tau_n)$ where $f : \mathcal{M}^n \mapsto \mathcal{M}$ is a multivariate polynomial function on n variables and $\tau_i \in \{0,1\}^*$ is the label of the i-th variable input of f.*

Notably, the data may be encrypted using different public keys, provided the decryptor has access to the respective private keys, usually derived from a master key. This is referred to as the *multi-user* labeled homomorphic encryption scheme, and the generalization is straightforward, but in this chapter, we only consider the symmetric version. We describe the scheme with the Joye-Libert cryptosystem serving as the additively homomorphic cryptosystem and adopt a definition largely similar to Barbosa et al. [147], with the tuple (KeyGen, Enc, Eval, Dec) defined as:

- KeyGen(1^λ): Run $\hat{\text{KeyGen}}(1^\lambda)$ to get (pk, sk'). Next, choose a random seed $K \in \{0,1\}^k$ for the pseudorandom function F, and set $\mathcal{L} = \{0,1\}^*$. Output sk = (sk', K) and epk = (pk, $\mathcal{L}$). The plaintext space $\mathcal{M}$ and the ciphertext space $\hat{\mathcal{C}}$ are given by the Joye-Libert cryptosystem.
- Enc(sk, τ, m): The encryption algorithm is composed of an offline and an online component. The argument sk is used to seed F.
  Offline-Enc(sk, τ): Compute $b \leftarrow F(\mathrm{K}, \tau)$ and $\beta \leftarrow \hat{\text{ENC}}(\text{pk}, b)$. Return $C_{\text{off}} = (b, \beta)$.
  Online-Enc(C_{off}): Return $C = (m - b, \beta) = (a, \beta) \in \mathcal{M} \times \hat{\mathcal{C}}$.
- Eval(epk, $f, C_1, \ldots, C_t, c_1, \ldots, c_s$): The evaluation function takes the public evaluation key epk, a multivariate polynomial f, ciphertexts $C_1, \ldots, C_t \in \mathcal{M} \times \hat{\mathcal{C}}$, and ciphertexts $c_1, \ldots, c_s \in \hat{\mathcal{C}}$, and is composed of four procedures.
  Mult:
  Mult(C_1, C_2): Return $\alpha = \hat{\text{ENC}}(pk, \mathrm{a}_1 \cdot \mathrm{a}_2) \boxplus (\mathrm{a}_1 \odot \beta_2) \boxplus (a_2 \odot \beta_1) \in \hat{\mathcal{C}}$.

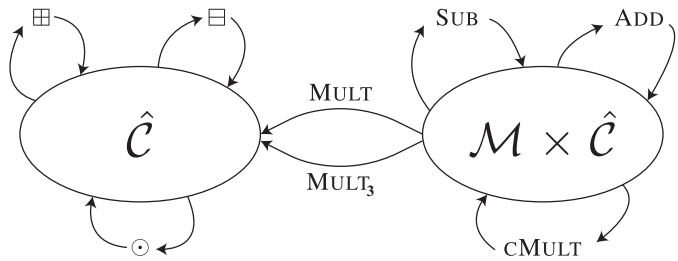


Figure 4.2 Relationship between the homomorphic operations and the ciphertext spaces.

MULT$_3(C_1, C_2, C_3, \beta_{1,2}, \beta_{1,3}, \beta_{2,3})$: Return $\alpha = \hat{\text{ENC}}(pk, \mathrm{a}_1 \cdot \mathrm{a}_2 \cdot \mathrm{a}_3) \boxplus \mathrm{a}_1 \odot \beta_{23} \boxplus \mathrm{a}_2 \odot \beta_{13} \boxplus \mathrm{a}_3 \odot \beta_{12} \boxplus (\mathrm{a}_1 \cdot \mathrm{a}_2) \odot \beta_3 \boxplus (\mathrm{a}_1 \odot \mathrm{a}_3) \cdot \beta_2 \boxplus (\mathrm{a}_2 \cdot \mathrm{a}_3) \odot \beta_1 \in \hat{\mathcal{C}}$.

ADD:

ADD(C_1, C_2): Return $C = (a = a_1 + a_2, \beta = \beta_1 \boxplus \beta_2)$.

ADD(c_1, c_2): Return $c = c_1 \boxplus c_2$.

SUB:

SUB(C_1, C_2): Return $C = (a = a_1 - a_2, \beta = \beta_1 \boxminus \beta_2)$.

SUB(c_1, c_2): Return $c = c_1 \boxminus c_2$.

CMULT:

CMULT(m, C): Return $C = (a = m \cdot a, \beta = m \odot \beta)$.

CMULT(m, c): Return $c = m \odot c$.

- DEC(sk, $\mathcal{P}$, C): The decryption function is composed of an offline and an online component.

  OFFLINE-DEC$(\text{sk}, \mathcal{P})$: Given sk and a labeled program $\mathcal{P} = (f, \tau_1, \ldots, \tau_t)$, the algorithm computes $b_i \leftarrow F(K, \tau_i)$, $b = f(b_1, \ldots, b_t)$ and outputs $sk_{\mathcal{P}} = (sk, b)$.

  ONLINE-DEC:

  ONLINE-DEC(C): There are two options if $C \in \mathcal{M} \times \hat{\mathcal{C}}$.

  (i) Output $m = a + b$; (ii) Output $m = a + \hat{\text{DEC}}(sk, \beta)$.

  ONLINE-DEC(c): If $c \in \hat{\mathcal{C}}$, set $\hat{m} = \hat{\text{DEC}}(sk, c)$ and output $m = \hat{m} + b$.

We show the relation between the homomorphic operations and the ciphertext spaces in Fig. 4.2. Correctness of SUB operations follows immediately from $\mathcal{M}$ being an additive group and $\hat{\mathcal{C}}$ being a multiplicative group. Correctness of MULT$_3$ is shown by Alexandru and Pappas [65], and correctness of the other operations is shown by Barbosa et al. [147]. Note that Alexandru and Pappas [65] showed that multiplication could be performed with an arbitrary number of ciphertexts, at the cost of an exponential increase in data traffic, thus limiting the practicality of evaluating higher-order polynomials. The general formula is omitted here for brevity. Notice, however, that a ciphertext $c \in \hat{\mathcal{C}}$ cannot be multiplied with another ciphertext, while ADD, SUB, and CMULT operations can still be performed. As such, the output from a multiplication operation behaves much like a Joye-Libert ciphertext. When proceeding, we will denote ciphertexts in $\mathcal{M} \times \hat{\mathcal{C}}$ by an uppercase C. In this chapter, the stream cipher HC-128 [103] takes the role of the pseudorandom function F.

4.4 Encrypted control using labeled homomorphic encryption

By permitting homomorphic addition, multiplication with plaintext constants, and multiplication with ciphertexts, labeled homomorphic encryption can be used to construct fully encrypted control systems. We assume that the operator of the cloud infrastructure is *honest-but-curious*, or *semi-honest*, in that the cloud infrastructure will seek to gain knowledge about the data but will not tamper with the incoming data or the uploaded code [13]. The reasoning behind this assumption is that reliable external computation is key to the value proposition of cloud computing services. However, a leak of possibly confidential information, such as system state information or control parameters, is much harder to detect. Consequently, maintaining confidentiality through all operations on the cloud infrastructure is a reasonable goal.

4.4.1 Fixed-point arithmetic

Since the plaintext space $\mathcal{M}$ of the labeled homomorphic encryption scheme is the commutative ring of integers $\mathbb{Z}_{2^k} = \{0, 1, \ldots, 2^k - 1\}$, and state estimates, control parameters, and control outputs are usually from some real interval $\mathcal{I} = [a, b] \subset \mathbb{R}$ represented with floating-point numbers, we need to map the real values to valid plaintexts. We define the map $\rho : \mathcal{I} \times \mathbb{Z}_{2^k} \mapsto \mathbb{Z}_{2^k}$ as

$$\rho(x, \gamma) = \begin{cases} \lceil \gamma x \rfloor, & \text{if } x \geq 0 \\ \lceil \gamma x \rfloor + 2^k, & \text{otherwise,} \end{cases} \tag{4.3}$$

where γ is a scaling constant and $\lceil \cdot \rfloor$ denotes rounding to the nearest integer. This mapping results in a rounding error

$$\epsilon \leq \frac{b - a}{2^{k+1}}, \tag{4.4}$$

assuming all values of the plaintext space are used. When ciphertexts are multiplied with l other ciphertexts or with plaintext constants, we need to consider the resulting cumulative scaling

$$m = \prod_{i=1}^{l} m_i = \prod_{i=1}^{l} \gamma_i x_i,$$

where

$$\gamma = \prod_{i=1}^{l} \gamma_i \leq \frac{2^k}{b - a} \tag{4.5}$$

must hold, a limitation previously discussed in literature concerning encrypted dynamic controllers where this limitation limits the lifespan of the control parameters [145, 157]. These constraints also give some flexibility to the system designer, where increased precision of certain encrypted variables must be weighed at the cost of reduced precision of other encrypted variables. Finally, once the resulting ciphertext

has been decrypted, we must map the recovered plaintext $m \in \mathcal{M}$ back to a real value $u \in \mathcal{I}$. We use the following mapping

$$\rho(m,\gamma)^{-1} = \begin{cases} \frac{m-2^k}{\gamma}, & \text{if } m \geq 2^{k-1} \\ \frac{m}{\gamma}, & \text{otherwise.} \end{cases} \tag{4.6}$$

Notably, this means that the *cumulative scaling* of each term that went into m must be the same. A practical example of this follows in Section 4.5.

4.4.2 Encrypted control using labeled homomorphic encryption

General discrete-time feedback controllers can be formulated as dynamical systems of the form

$$x[k+1] = Ax[k] + Bv[k] \tag{4.7}$$
$$u[k] = Cx[k] + Dv[k], \tag{4.8}$$

where x is the state of the controller, v is the input to the controller, u is the output of the controller, and the matrices A, B, C, and D are the parameters of the controller. Generally, designing encrypted variants of such controllers is problematic because homomorphic multiplications cause a cumulative scaling in the controller state, eventually leading to overflow or underflow problems. However, special forms of (4.7) where the controller dynamics matrix A does not require scaling do not suffer from the cumulative scaling problem. An important example of such a controller is the PID controller, which is still the most common dynamic feedback control system used by industry [158]. A discrete-time state-space PID controller with feed-forward can be written as

$$x[k+1] = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e[k-1] \\ i[k] \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} r[k] \\ e[k] \end{bmatrix} \tag{4.9}$$

$$u[k] = \begin{bmatrix} -\frac{K_d}{\Delta t} & K_i \end{bmatrix} \begin{bmatrix} e[k-1] \\ i[k] \end{bmatrix} + \begin{bmatrix} K_{\text{ff}} & \frac{K_d}{\Delta t} \end{bmatrix} \begin{bmatrix} r[k] \\ e[k] \end{bmatrix}, \tag{4.10}$$

where e denotes the state error, i denotes the integrator state, and r denotes the input reference, assuming the timestep Δt is significantly smaller than the time constant of the dynamics of the system. The control parameters can then be mapped to plaintext space using (4.3) and encrypted. The Mult operation handles the feed-forward and proportional terms, while the derivative control term can be computed by using Mult_3. For the integral term, we have to take care that the terms are multiplied in the same operation and summed afterward, that is, rewritten as

$$\sum_{i=0}^{k} K_{\text{i}} e[k] \Delta t \tag{4.11}$$

With this design, the integral term can be seen as a recursive homomorphic addition of products of three ciphertexts. By design, integral action makes the encrypted controller *stateful*. Since the decryption algorithm needs to know the exact operations and data with which the ciphertext has been produced to successfully

decrypt, this means that *reliable data transfer* between the USV and the cloud controller is also necessary. A possible work-around to make the encrypted controller stateless is to encrypt and transmit the cumulative error $\sum_{i=0}^{k} e[k]\Delta t$ to the cloud controller, after which the cumulative error only needs to be multiplied with the integral gain.

We can now use the operations available from the labeled homomorphic encryption scheme and formulate an encrypted form of the PID controller as

$$c_{\text{integral}}[0] = \textsc{Enc}(\text{sk}, \tau_{\text{integral},\,0}, 0) \tag{4.12}$$

$$\begin{aligned} c_{\text{integral}}[k] &= c_{\text{integral}}[k-1] \\ &\quad \boxplus \textsc{Mult}_3(C_{K_\text{i}}, C_{e[k]}, C_{\Delta t}, \\ &\qquad \beta_{K_\text{i} e[k]}, \beta_{K_\text{i}\Delta t}, \beta_{e[k]\Delta t}) \end{aligned} \tag{4.13}$$

$$\begin{aligned} c_{u[k]} &= \textsc{Mult}(C_{K_\text{ff}}, C_{r[k]}) \\ &\quad \boxplus \textsc{Mult}(C_{K_\text{p}}, C_{e[k]}) \\ &\quad \boxplus \textsc{Mult}_3(C_{K_\text{d}}, \textsc{Sub}(C_{e[k]}, C_{e[k-1]}), C_{\Delta t^{-1}}, \\ &\qquad \beta_{K_\text{d} e[k]}, \beta_{K_\text{d}\Delta t^{-1}}, \beta_{e[k]\Delta t^{-1}}) \\ &\quad \boxplus c_{\text{integral}}[k] \end{aligned} \tag{4.14}$$

Throughout these computations, we must ensure that the cumulative scaling satisfies (4.5) and is similar for each term such that (4.6) successfully recovers the control signal, that is,

$$\begin{aligned} \gamma &= \gamma_{K_\text{ff}} \gamma_r \\ &= \gamma_{K_p} \gamma_e \\ &= \gamma_{K_d} \gamma_e \gamma_{\Delta t} \\ &= \gamma_{K_i} \gamma_e \gamma_{\Delta t} \\ &\leq \frac{2^k}{b-a}. \end{aligned} \tag{4.15}$$

The code and sample programs demonstrating the described scheme are available in [159].

4.4.3 Choosing k and n

The choice of k and n affects the size of the plaintext space, and hence the rounding error induced by (4.3), and the *security* of the cryptosystem, respectively. Natural choices for k in software implementations are $k = 32$ and $k = 64$, such that plaintexts can be represented by 4-byte or 8-byte unsigned integers, respectively. We note here that the Joye-Libert decryption algorithm recovers each bit of the plaintext individually. Therefore, we expect the computational latency induced by Alg. 7 to grow linearly with k, which is asymptotically slower than the Paillier cryptosystem. Despite this, we believe the Joye-Libert cryptosystem is a reasonable choice since the ciphertext expansion of the Joye-Libert cryptosystem is only half of the Paillier cryptosystem, and several operations involve multiple multiplications in ciphertext space. Examples of relevant choices for n are $n = 2048$ and $n = 3072$, corresponding

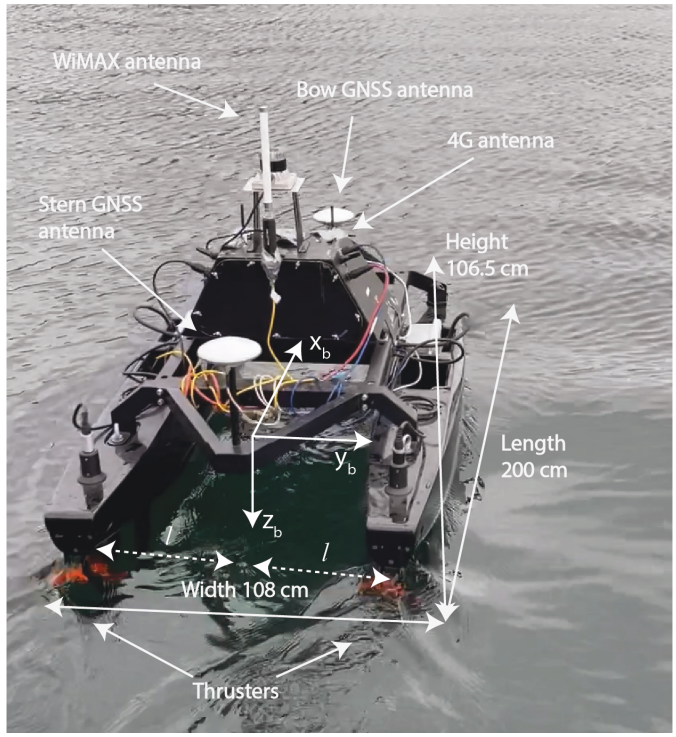


Figure 4.3 An illustration of the Cyberotter USV.

to approximately 112 and 128-bit security against brute-force attacks, respectively, the first of which is considered secure until at least the year 2030, while the latter is considered secure for the foreseeable future [160].

4.5 Case study: Encrypted surge speed and yaw control of USV

We proceed with a case study where we design and implement an encrypted control system for the Cyberotter USV, shown in Fig. 4.3. The USV is actuated by two fixed thrusters mounted at the stern on the starboard and port hulls. As a result, the USV is underactuated and can only control yaw and surge. The surge speed is controlled by setting a common thrust, while the yaw is controlled by setting a differential thrust. The desired surge speed is fixed on each straight-line segment between waypoints, while an ILOS guidance system [126] produces the desired yaw. The error in surge speed and yaw are mapped to valid plaintexts and encrypted before it is sent to the cloud. Upon reception, the encrypted surge speed controller computes an encrypted common thrust, while the encrypted yaw controller computes an encrypted differential thrust. Homomorphically adding and subtracting the encrypted differential thrust to the encrypted common thrust yields the encrypted thrust allocation to the starboard and port thrusters, respectively.

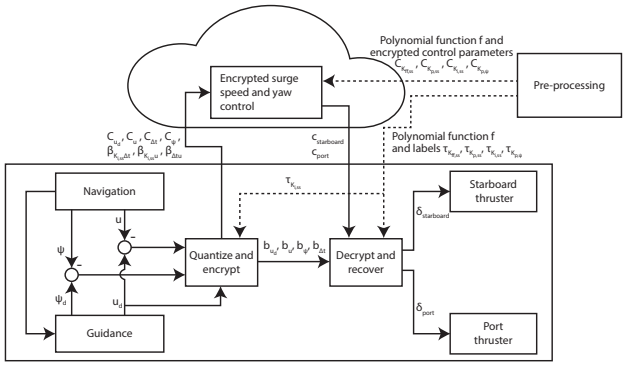


Figure 4.4 An illustration of signal flow in the encrypted surge speed and yaw controller.

The USV is port-starboard symmetric, implying that the surge is decoupled from the sway and yaw. Therefore, we can design independent controllers for the surge and yaw.

We consider a 3 degrees of freedom maneuvering model of the form [81, p. 157]

$$\dot{\boldsymbol{\eta}} = \boldsymbol{R}_b^n(\psi)\boldsymbol{\nu}$$
$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{N}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} \tau_1 \\ 0 \\ \tau_3 \end{bmatrix},$$

where $\boldsymbol{\eta} = [N, E, \psi]^T \in \mathbb{R}^2 \times \mathbb{S}$ describes the vehicle pose in the earth-fixed NED reference frame, $\boldsymbol{\nu} = [u, v, r]^T \in \mathbb{R}^3$ describes the vehicle velocity in the the body-fixed frame, $\boldsymbol{R}_b^n(\psi) \in \mathrm{SO}(3)$ is a rotation matrix from the body-fixed frame to the NED frame, $\boldsymbol{M}$ is the mass-inertial matrix, and $\boldsymbol{N}(\boldsymbol{\nu})$ describes the Coriolis, centripetal, and damping forces. Moreover, we have

$$\begin{bmatrix} \tau_1 \\ 0 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} K(\delta_{\text{port}} + \delta_{\text{starboard}}) \\ 0 \\ Kl(\delta_{\text{port}} - \delta_{\text{starboard}}), \end{bmatrix} \tag{4.16}$$

where K denotes the propeller thrust coefficient, l denotes the unsigned lever arm from the body x-axis to the thrusters, and $\delta_{\text{port}} = \frac{1}{2}(\delta_{\text{ss}} + \delta_\psi)$ and $\delta_{\text{starboard}} = \frac{1}{2}(\delta_{\text{ss}} - \delta_\psi)$. After defining $\tilde{u} = u_d - u$ and $\tilde{\psi} = \psi_d - \psi$, where u_d and ψ_d are the desired surge speed and yaw, respectively, we can compute δ_{ss} and δ_ψ according to the following linear control laws

$$\delta_{\text{ss}} = K_{\text{ff, ss}} u_d[k] + K_{\text{p, ss}} \tilde{u}[k] + K_{\text{i, ss}} \sum_{i=1}^{k} \tilde{u}[k]\Delta t \tag{4.17}$$

$$\delta_\psi = K_{\text{p, }\psi} \tilde{\psi}[k], \tag{4.18}$$

where (4.17) and (4.18) represent the surge speed and yaw controllers, respectively. Here, we have assumed that the control loop is significantly faster than the dynamics

of the system, which we believe is a reasonable assumption for a USV. The goal is to implement these in encrypted form using (4.12), (4.13), and (4.14). Notice that the ILOS guidance system and the yaw controller can be seen as two controllers in a cascade. With integral action in the outer ILOS guidance system, we avoid using integral action in the inner yaw control loop, which would reduce the phase margin and hence the overall stability of the yaw control action.

The first step is to map the control parameters $K_{\mathrm{ff,\,ss}}, K_{\mathrm{p,\,ss}}, K_{\mathrm{i,\,ss}}$, and $K_{\mathrm{p},\,\psi}$ to suitable plaintexts. The corresponding plaintexts can then be encrypted to obtain the encrypted control parameters, which must be uploaded to the cloud along with the polynomial to be evaluated. Moreover, the labels associated with the encrypted control parameters must be uploaded to the USV for successful encryption and decryption. Pseudocode for this step is given by Alg. 8.

Algorithm 8 Offline preprocessing

Parameters

sk, K	Secret key
$\gamma_{K_{\mathrm{ff,\,ss}}}, \gamma_{K_{\mathrm{p,\,ss}}}$	
$\gamma_{K_{\mathrm{i,\,ss}}}, \gamma_{K_{\mathrm{p},\,\psi}}$	Scaling factors

Input

$K_{\mathrm{ff,\,ss}}$	Surge speed feed-forward gain
$K_{\mathrm{p,\,ss}}$	Surge speed proportional gain
$K_{\mathrm{i,\,ss}}$	Surge speed integral gain
$K_{\mathrm{p},\,\psi}$	Yaw proportional gain

Output

$C_{K_{\mathrm{ff,\,ss}}}$	Encrypted surge speed feed-forward gain
$C_{K_{\mathrm{p,\,ss}}}$	Encrypted surge speed proportional gain
$C_{K_{\mathrm{i,\,ss}}}$	Encrypted surge speed integral gain
$C_{K_{\mathrm{p},\,\psi}}$	Encrypted yaw proportional gain
$b_{K_{\mathrm{ff,\,ss}}}$	Output of $F(K, \tau_{K_{\mathrm{ff,\,ss}}})$
$b_{K_{\mathrm{p,\,ss}}}$	Output of $F(K, \tau_{K_{\mathrm{p,\,ss}}})$
$b_{K_{\mathrm{i,\,ss}}}$	Output of $F(K, \tau_{K_{\mathrm{i,\,ss}}})$
$b_{K_{\mathrm{p},\,\psi}}$	Output of $F(K, \tau_{K_{\mathrm{p},\,\psi}})$

1: **function** OfflinePreprocessing
2: $\bar{K}_{\mathrm{ff,\,ss}} \leftarrow \rho(K_{\mathrm{ff,\,ss}}, \gamma_{K_{\mathrm{ff,\,ss}}})$, $\bar{K}_{\mathrm{p,\,ss}} \leftarrow \rho(K_{\mathrm{p,\,ss}}, \gamma_{K_{\mathrm{p,\,ss}}})$,
$\bar{K}_{\mathrm{i,\,ss}} \leftarrow \rho(K_{\mathrm{i,\,ss}}, \gamma_{K_{\mathrm{i,\,ss}}})$, $\bar{K}_{\mathrm{p},\,\psi} \leftarrow \rho(K_{\mathrm{i},\,\psi}, \gamma_{K_{\mathrm{p},\,\psi}})$
3: Choose unique labels $\tau_{K_{\mathrm{ff,\,ss}}}, \tau_{K_{\mathrm{p,\,ss}}}, \tau_{K_{\mathrm{i,\,ss}}}, \tau_{K_{\mathrm{p},\,\psi}} \in \mathcal{L}$
4: $C_{K_{\mathrm{ff,\,ss}}} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{K_{\mathrm{ff,\,ss}}}, \bar{K}_{\mathrm{ff,\,ss}})$
5: $C_{K_{\mathrm{p,\,ss}}} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{K_{\mathrm{p,\,ss}}}, \bar{K}_{\mathrm{p,\,ss}})$
6: $C_{K_{\mathrm{i,\,ss}}} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{K_{\mathrm{i,\,ss}}}, \bar{K}_{\mathrm{i,\,ss}})$
7: $C_{K_{\mathrm{p},\,\psi}} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{K_{\mathrm{p},\,\psi}}, \bar{K}_{\mathrm{p},\,\psi})$
8: Send $[C_{K_{\mathrm{ff,\,ss}}}, C_{K_{\mathrm{p,\,ss}}}, C_{K_{\mathrm{i,\,ss}}}, C_{K_{\mathrm{p},\,\psi}}]$ to the cloud
9: Send $[\tau_{K_{\mathrm{ff,\,ss}}}, \tau_{K_{\mathrm{p,\,ss}}}, \tau_{K_{\mathrm{i,\,ss}}}, \tau_{K_{\mathrm{p},\,\psi}}]$ to the decryption device
10: Send $\tau_{K_{\mathrm{i,\,ss}}}$ to the encryption device
11: **end function**

Onboard the USV, the desired surge speed, error in surge speed, error in yaw,

and timestep must be mapped to appropriate plaintexts before they are encrypted. The Joye-Libert cryptosystem must be used to explicitly encrypt additional data required to perform multiplication with three ciphertexts. The ciphertexts and the additional data are then transmitted to the cloud controller, and the labels used are sent to the decryption algorithm. Pseudocode for the mapping and encryption algorithm is shown in Alg. 9.

Algorithm 9 Mapping and encryption on-board USV

Parameters

Q	Queue to pass b's
sk, K	Secret key
$\tau_{K_{\mathrm{i,\ ss}}}$	Label associated with integral gain
γ_{u_d}, γ_u	
$\gamma_\psi, \gamma_{\Delta t}$	Scaling factors

Input

u	Surge speed
u_d	Desired surge speed
ψ	Yaw
ψ_d	Desired yaw
Δt	Timestep

Output

C_{u_d}	Encrypted desired surge speed
C_u	Encrypted error in surge speed
$C_{\Delta t}$	Encrypted timestep
C_ψ	Encrypted error in yaw
$\beta_{K_{\mathrm{i,\ ss}}\Delta t}$	Encrypted product of $b_{K_{\mathrm{i,\ ss}}}$ and $b_{\Delta t}$
$\beta_{K_{\mathrm{i,\ ss}}u}$	Encrypted product of $b_{K_{\mathrm{i,\ ss}}}$ and b_u
$\beta_{\Delta t u}$	Encrypted product of $b_{\Delta t}$ and b_u

1: **function** QuantizeAndEncrypt
2: Choose unique labels $\tau_{u_d}, \tau_u, \tau_\psi, \tau_{\Delta t} \in \mathcal{L}$
3: $\bar{u}_d \leftarrow \rho(u_d, \gamma_{u_d})$, $\bar{u} \leftarrow \rho(u_d - u, \gamma_u)$,
$\bar{\psi} \leftarrow \rho(\psi_d - \psi, \gamma_\psi)$, $\bar{\Delta t} \leftarrow \rho(\Delta t, \gamma_{\Delta t})$
4: $C_{u_d} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{u_d}, \bar{u}_d)$
5: $C_u \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{\bar{u}}, \bar{u})$
6: $C_\psi \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_\psi, \bar{\psi})$
7: $C_{\Delta t} \leftarrow \mathrm{Enc}(\mathrm{sk}, \tau_{\Delta t}, \bar{\Delta t})$
8: $b_u \leftarrow F(K, \tau_{b_u})$, $b_{\Delta t} \leftarrow F(K, \tau_{\Delta t})$
$b_{K_{\mathrm{i,\ ss}}} \leftarrow F(K, \tau_{K_{\mathrm{i,\ ss}}})$
9: $\beta_{K_{\mathrm{i,\ ss}}\Delta t} \leftarrow \hat{\mathrm{Enc}}(b_{K_{\mathrm{i,\ ss}}} \cdot b_{\Delta t})$
10: $\beta_{K_{\mathrm{i,\ ss}}u} \leftarrow \hat{\mathrm{Enc}}(b_{K_{\mathrm{i,\ ss}}} \cdot b_u)$
11: $\beta_{\Delta t u} \leftarrow \hat{\mathrm{Enc}}(b_{\Delta t} \cdot b_u)$
12: Q.Enqueue(b_{u_d}); Q.Enqueue(b_u);
Q.Enqueue(b_ψ); Q.Enqueue($b_{\Delta t}$)
13: Send $[C_{u_d}, C_u, C_{\Delta t}, C_\psi, \beta_{K_{\mathrm{i,\ ss}}\Delta t}, \beta_{K_{\mathrm{i,\ ss}}u}, \beta_{\Delta t u}]$ to the cloud
14: **end function**

Upon reception of the encrypted data from the USV, the encrypted surge speed and yaw controller in the cloud evaluates the uploaded polynomial function and sends the encrypted port and starboard thrust allocations back to the USV. Pseudocode for the encrypted surge speed and yaw controller is shown in Alg. 10.

Algorithm 10 Encrypted surge speed and yaw controller

Parameters

epk	Public evaluation key
$C_{K_{\text{ff, ss}}}$	Encrypted surge speed feed-forward gain
$C_{K_{\text{p, ss}}}$	Encrypted surge speed proportional gain
$C_{K_{\text{i, ss}}}$	Encrypted surge speed integral gain
$C_{K_{\text{p, }\psi}}$	Encrypted yaw proportional gain
$c_{\text{integral, 0}}$	Encrypted surge speed integral state

Input

C_{u_d}	Encrypted desired surge speed
C_u	Encrypted error in surge speed
$C_{\Delta t}$	Encrypted timestep
C_ψ	Encrypted error in yaw
$\beta_{K_{\text{i, ss}}\Delta t}$	Encrypted product of $b_{K_{\text{i, ss}}}$ and $b_{\Delta t}$
$\beta_{K_{\text{i, ss}}u}$	Encrypted product of $b_{K_{\text{i, ss}}}$ and b_u
$\beta_{\Delta t u}$	Encrypted product of $b_{\Delta t}$ and b_u

Output

c_{port}	Encrypted port thrust allocation
$c_{\text{starboard}}$	Encrypted starboard thrust allocation

function EncryptedCloudController
  $c_{\text{integral}} \leftarrow c_{\text{integral, 0}}$
  for each new message $[C_{u_d}, C_u, C_{\Delta t}, C_\psi, \beta_{K_{\text{i, ss}}\Delta t}, \beta_{K_{\text{i, ss}}u}, \beta_{\Delta t u}]$ **do**
    $c_{\text{ss}} \leftarrow \text{Add}(\text{Mult}(C_{K_{\text{ff, ss}}}, C_{u_d}),\ \text{Mult}(C_{K_{\text{p, ss}}}, C_u))$
    $c_{\text{integral}} \leftarrow \text{Add}(c_{\text{integral}}, \text{Mult}_3(C_{K_{\text{i, ss}}}, C_u, C_{\Delta t}, \beta_{K_{\text{i, ss}}\Delta t}, \beta_{K_{\text{i, ss}}u}, \beta_{\Delta t u}))$
    $c_{\text{ss}} \leftarrow \text{Add}(c_{\text{ss}}, c_{\text{integral}})$
    $c_\psi \leftarrow \text{Mult}(C_{K_{\text{p, }\psi}}, C_\psi)$
    $c_{\text{port}} \leftarrow \text{Add}(c_{\text{ss}},\ c_\psi)$
    $c_{\text{starboard}} \leftarrow \text{Sub}(c_{\text{ss}},\ c_\psi)$
    Output $[c_{\text{port}}, c_{\text{starboard}}]$
  end for
end function

Upon reception of the encrypted control signals, the decryption algorithm evaluates the polynomial f over the labels of the data that went into producing the ciphertexts, that is, $b = f(b_1, \ldots, b_t)$. It then computes Dec-Offline and Dec-Online to recover the appropriate plaintexts, after which the port and starboard thrust allocations are recovered using (4.6). Pseudocode for the decryption algorithm is shown in Alg. 11. An illustration of the system architecture and the signal flow between the components is shown in Fig. 4.4.

Regarding the choice of the pseudorandom function F, we use the stream cipher HC-128, which is stateful by definition. While HC-128 is very efficient once initialized, the initialization overhead is significant. Therefore, it would be more prudent to choose a pseudorandom function with a smaller initialization overhead if reliable communication is not used since re-initialization would be required more frequently because of, for example, packet loss. Suitable options under such circumstances include a block cipher, for example, AES [88], in Counter mode, or a cryptographic hash function with a counter, for example, SHA-3 [161].

Algorithm 11 Recovery of encrypted thrust allocation

Parameters

Q	Queue to receive b's
sk, K	Secret key
b_{integral}	Surge speed integral label state
γ	Cumulative scaling factor

Input

c_{port}	Encrypted port thrust allocation
$c_{\text{starboard}}$	Encrypted starboard thrust allocation

Output

δ_{port}	Port thrust allocation
$\delta_{\text{starboard}}$	Starboard thrust allocation

1: **function** DecryptAndRecover
2: $b_{\text{integral}} \leftarrow b_{\text{integral, 0}}$
3: **for** each new message $[c_{\text{port}}, c_{\text{starboard}}]$ **do**
4: $b_{u_d} \leftarrow$ Q.Dequeue(); $b_u \leftarrow$ Q.Dequeue();
$b_\psi \leftarrow$ Q.Dequeue(); $b_{\Delta t} \leftarrow$ Q.Dequeue()
5: $b_{\text{integral}} \leftarrow b_{\text{integral}} + b_{K_{\text{i,ss}}} \cdot b_u \cdot b_{\Delta t}$
6: $b_{\text{ss}} \leftarrow b_{K_{\text{ff,ss}}} \cdot b_{u_d} + b_{K_{\text{p,ss}}} \cdot b_u + b_{\text{integral}}$
7: $b_\psi \leftarrow b_{K_{\text{p, }\psi}} \cdot b_\psi$
8: $b_{\text{port}} \leftarrow b_{\text{ss}} + b_\psi$
9: $b_{\text{starboard}} \leftarrow b_{\text{ss}} - b_\psi$
10: $\bar{\delta}_{\text{port}} \leftarrow \text{Dec}(\text{sk}, b_{\text{port}}, c_{\text{port}})$
11: $\bar{\delta}_{\text{starboard}} \leftarrow \text{Dec}(\text{sk}, b_{\text{starboard}}, c_{\text{starboard}})$
12: $\delta_{\text{port}} \leftarrow \rho^{-1}(\bar{\delta}_{\text{port}}, \gamma)$
13: $\delta_{\text{starboard}} \leftarrow \rho^{-1}(\bar{\delta}_{\text{starboard}}, \gamma)$
14: Output $[\delta_{\text{port}}, \delta_{\text{starboard}}]$
15: **end for**
16: **end function**

4.5.1 Computational latency

The Joye-Libert cryptosystem, the labeled homomorphic encryption scheme, and the encrypted controllers were implemented in C++ using number-theoretic functions and big-number representation from the GNU Multiple Precision Arithmetic Library [162]. We use the HC-128 implementation described by [79]. The computational latency induced by the cryptographic operations is of significant importance since it

Table 4.1 System specifications for the experiments

Hardware	
Model name	NVIDIA Jetson Xavier
CPU	NVIDIA Tegra Xavier
Instruction set architecture	ARMv8.2
Number of cores	8
Word size	64 bit
Memory	32GB
Software	
Operating system	Ubuntu 18.04 LTS
Compiler	g++ 7.5.0

limits the frequency of the control loop, and many dynamical systems must satisfy real-time constraints. Figure 4.5 shows the computational latencies of Algs. 9, 10, and 11 when executed with 32-bit and 64-bit plaintexts and a 2048-bit and 3072-bit factoring modulus. We performed the tests on an Nvidia Jetson Xavier, and the system specifications are summarized in Table 4.1.

The increased computational latency of the decryption algorithm when the size of the plaintext is increased is only marginal, with the majority of the increased computational latency occurring in the encrypted controller algorithm. This is a reasonable result, considering the computational cost of the MULT and MULT_3 operations, yet it is also a result in favor of the Joye-Libert cryptosystem over the Paillier cryptosystem as part of the labeled homomorphic encryption scheme. Additionally, we find that our system produces much lower computational latencies than the system proposed by Teranishi et al. [151], built using the multiplicatively homomorphic Elgamal cryptosystem, despite being tested on a significantly less powerful platform. This result might also be influenced by performance differences between the big-number libraries used in the respective implementations.

We note that the induced latency is significantly higher than the latency induced by the scheme proposed by Cheon et al. [73]. However, that scheme is fundamentally different as it only allows evaluation of linear functions, that is, semi-encrypted control systems, and does not permit multiplication of ciphertexts as our scheme and the scheme by Teranishi et al. [151] does.

4.6 Experimental validation through field experiments

Field experiments of the encrypted cloud-based surge speed and yaw controller were conducted in the Trondheim Fjord using the Cyberotter USV. The parameters $k = 64$ and $n = 3072$ were used, and we assume that the corresponding plaintext value of all ciphertexts falls in the interval $\mathcal{I} = [-2000, 2000]$. According to (4.15), we must then pick scaling factors such that $\gamma \leq \frac{2^{64}}{4000} \approx \frac{2^{64}}{2^{12}} = 2^{52}$ to prevent overflow. Additionally, we know that $\psi \in [-\pi, \pi]$ and assume that $u, u_d \in [-3, 3]$ and $\Delta t \in [0, 1]$. The control parameters used are $K_{\text{ff, ss}} = 100, K_{\text{p, ss}} = 450, K_{\text{i, ss}} = 25$, and

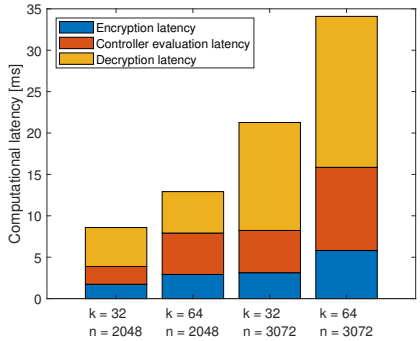


Figure 4.5 Computational latencies with k-bit plaintexts and n-bit factoring modulus.

$K_{\mathrm{p},\,\psi} = 1$. As such we set $\gamma = 2^{52}$, and choose the scaling factors as $\gamma_{u_d} = \gamma_\psi = 2^{52}$, $\gamma_u = \gamma_{\Delta t} = \gamma_{K_{\mathrm{p},\,\mathrm{ss}}} = 2^{26}$, and $\gamma_{K_{\mathrm{ff},\,\mathrm{ss}}} = \gamma_{K_{\mathrm{i},\,\mathrm{ss}}} = \gamma_{K_{\mathrm{p},\,\psi}} = 1$. With these choices, (4.4) gives rounding errors of $\epsilon_u \leq \frac{6}{2\cdot 2^{26}} \approx 4.47 \times 10^{-8}$, $\epsilon_{u_d} \leq \frac{6}{2\cdot 2^{52}} \approx 6.66 \times 10^{-16}$, $\epsilon_\psi \leq \frac{2\pi}{2\cdot 2^{52}} \approx 6.98 \times 10^{-16}$, and $\epsilon_{\Delta t} \leq \frac{1}{2\cdot 2^{26}} \approx 7.45 \times 10^{-9}$. Clearly, these rounding errors are too insignificant to cause any measurable performance degradation in the system.

4.6.1 Experimental setup

In the experiments, an Nvidia Jetson Xavier located in an on-shore office hosts the encrypted control system. An IPsec tunnel is established between two industrial 4G routers, one onboard the USV and another in the on-shore office, securing the transmitted data. We stress the importance of imposing additional security mechanisms on the communication link between the USV and the encrypted controller. The honest-but-curious assumption is only reasonable for the cloud infrastructure, not the communication links. Since the ciphertexts produced by homomorphic cryptosystems are malleable by design, it is critical to verify the authenticity of the data upon reception, for example, by using message authentication codes.

The mission objective for the USV is to follow a path with straight-line segments between waypoints while maintaining a fixed surge speed along each line segment. We performed three consecutive experiments for experimental validation, two of which were conducted on a stretch of water mostly sheltered from the prevailing winds and currents. The third experiment was performed on a more exposed stretch of water. Missions with waypoints and desired surge speeds on each straight-line segment between waypoints were uploaded from a laptop to the USV using a WiMAX link, and we monitored the performance of the USV from a recreational boat during each experiment.

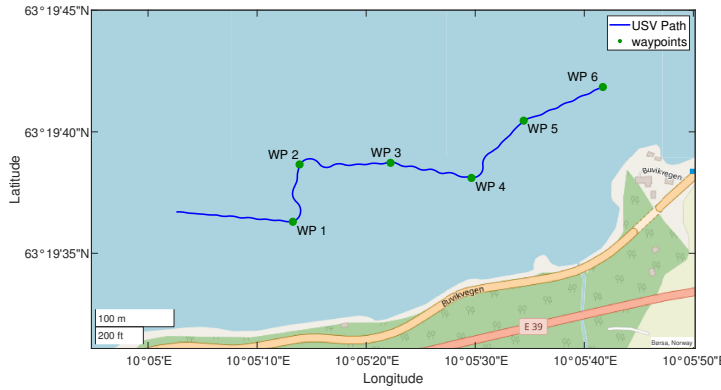


Figure 4.6 The path of the USV and the associated waypoints in Experiment 1.

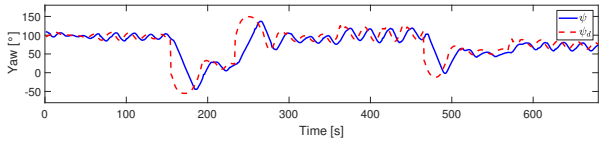
4.6.2 Results

We present the results by plotting the path of the USV against the waypoints. Additionally, we plot the yaw and surge speed of the USV against the desired yaw and speed. Note that the yaw and the desired yaw are presented as unconstrained angles to avoid discontinuities. This is done to enhance the clarity of the presentation.

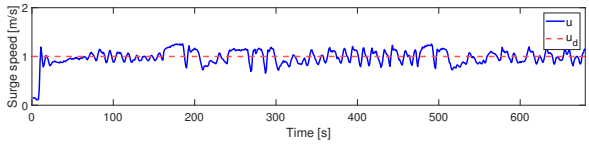
Figure 4.6 shows the waypoints and the path of the USV in Experiment 1, while Fig. 4.7a shows the yaw of the USV plotted against the desired yaw produced by the guidance system. The surge speed of the USV is plotted against the desired surge speed in Fig. 4.7b. Similarly, the path of the USV in Experiment 2 is shown in Fig. 4.8, and the yaw and surge speed are plotted against the desired yaw and the desired surge speed in Figs. 4.9a and 4.9b, respectively. Finally, Fig. 4.10 shows the path of the USV in Experiment 3, while the yaw and surge speed are plotted against the desired yaw and the desired surge speed in Figs. 4.11a and 4.11b, respectively. A link to video showing excerpts of Experiments 1 and 3 can be found in Appendix B.

4.7 Discussion

The results show that the USV successfully followed the desired path with the encrypted control system. We observed some oscillations in yaw, especially during the third experiment. This was likely caused by wind gusts, most notable during the third experiment, resulting in a rapidly shifting crab angle which caused oscillations in the desired yaw produced by the ILOS guidance system. Moreover, we note that we experienced some oscillations in the surge speed. We suspect the reason to be that the controller produced thruster revolutions per minutes (RPMs) as output. Since the thrust curve of the propellers is not symmetric, they produce more forward thrust than backward thrust for a given RPM. Therefore, the yaw control action resulted in a net-forward thrust, causing an increase in surge speed when the USV was turning. Ideally, the controller should have produced forward and differential thrust in newtons, which we should have mapped to appropriate



(a)



(b)

Figure 4.7 Results from Experiment 1. **(a)** The yaw of the USV plotted against the desired yaw. **(b)** The surge speed of the USV plotted against the desired surge speed.

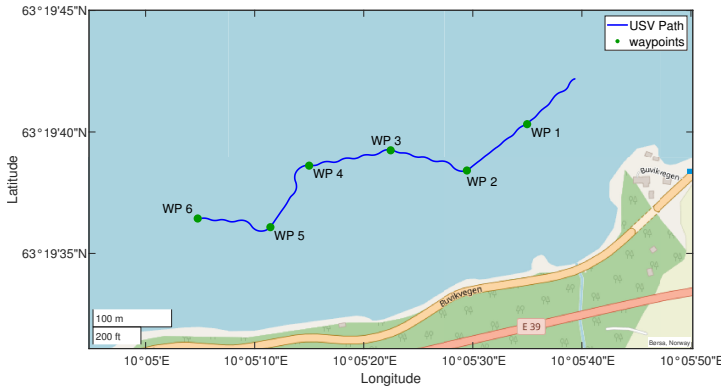
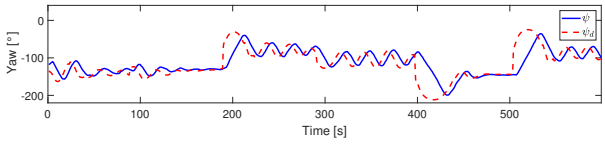
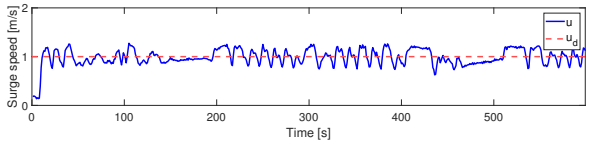


Figure 4.8 The path of the USV and the associated waypoints in Experiment 2.



(a)



(b)

Figure 4.9 Results from Experiment 2. **(a)** The yaw of the USV plotted against desired yaw. **(b)** The surge speed of the USV plotted against the desired surge speed.

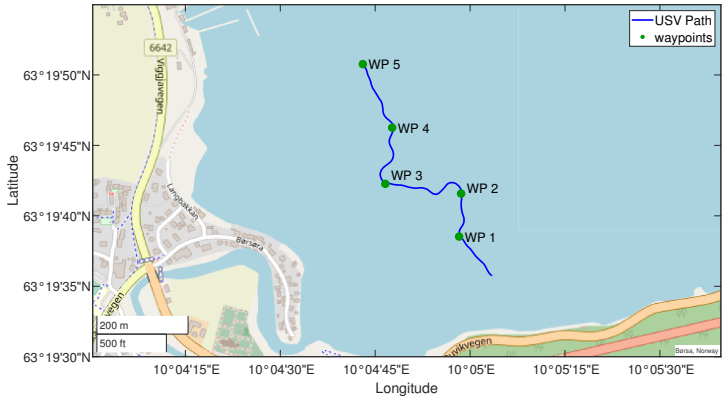


Figure 4.10 The path of the USV and the associated waypoints in Experiment 3.

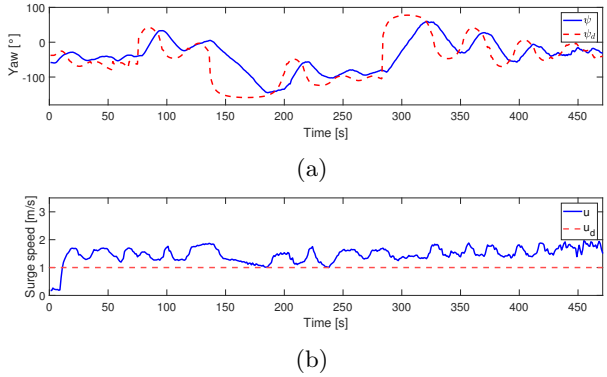


Figure 4.11 Results from Experiment 3. **(a)** The yaw of the USV plotted against the desired yaw. **(b)** The surge speed of the USV plotted against the desired surge speed.

RPMs. This is listed as future work.

From a practical perspective, we note that encrypted control systems have significant limitations compared to conventional control systems. A cryptosystem that allows logical comparisons between ciphertexts, for example, by finding that one ciphertext is 'greater' than another ciphertext, is severely compromised and is not semantically secure. This means it is impossible to implement encrypted nonlinear logic such as anti-windup and saturation elements, features frequently found in industrial control systems. Moreover, as noted earlier, the integral action results in a stateful controller. Since the decryption function requires exact knowledge of the polynomial evaluated, this forces us to use reliable data transfer. If we instead had sent the cumulative error, the encrypted controller would be stateless. However, in such a scenario, it might make more sense to use a multiplicatively homomorphic cryptosystem, such as Elgamal.

The transmission delay induced by the communication link with the cloud is

another important factor to consider, as transmission delays result in delayed action and a reduced phase margin. In our experiment, the round-trip communication delay ranged between 60 and 300 ms. A more consistent time delay would likely have been observed with an unreliable transport protocol, such as UDP. However, because of the slow dynamics of the USV, the performance degradation is not significant. The expected transmission delay of such systems is highly dependent on the spatial location of each module relative to the communication infrastructure and the communication infrastructure used. For example, we would expect a 5G connection to result in a reduced communication delay compared to the 4G connection used in our experiment.

In this chapter, we considered the use of the Joye-Libert cryptosystem. The motivation behind using the Joye-Libert cryptosystem over the Paillier cryptosystem is the reduction in the amount of data transmitted, resulting in a more cost-efficient implementation since mobile broadband is often charged on a pay-as-you-go basis or for a fixed amount of data. Figure 4.5 also showed that for a fixed-size modulus, most of the increase in computational latency for a given plaintext size k originated in the evaluation of the encrypted control algorithm and not in the decryption algorithm. As a result, using the Paillier cryptosystem instead of the Joye-Libert cryptosystem would result in a negligible reduction of the decryption latency, dwarfed by the likely increase in computational latency of the encrypted control algorithm because of the increased ciphertext expansion. There exist other generalizations of the Goldwasser-Micali cryptosystem, with a similar ciphertext expansion to the Joye-Libert cryptosystem, but with an asymptotically faster decryption algorithm, on par with Paillier, that we did not consider here. For example, we could have used the scheme proposed by Zhao et al. [163].

Potential users should also question the necessity of using encrypted control parameters in their systems. If the multi-user variant of the labeled homomorphic cryptosystem is used, the control parameters and the state information can be encrypted using unique keys. But by knowing the state information going in and the thrust allocations coming out of the encrypted control system, system identification methods can be used to identify the control parameters used. Therefore, the control parameters should only be considered unknown by the cloud operator. In any event, converting the proposed encrypted control system to a partially encrypted control system with plaintext control parameters is straightforward and eliminates the need to know the mathematical operations and variables involved in each ciphertext for successful decryption, greatly simplifying the overall system.

4.8 Chapter summary

In this chapter, we have shown how the labeled homomorphic encryption scheme can be used to design encrypted control systems. Moreover, as a case study, we developed an encrypted control system for a USV and demonstrated its efficiency through an extensive field experiment in a challenging environment. We found that the proposed method worked very well, with reduced computational latencies compared to a previously proposed encrypted controller, and is well suited for marine crafts characterized by slow dynamics. Despite this, potential users should be aware

that nonlinear logic frequently used in feedback control, such as anti-windup and saturation elements, is not available in encrypted control systems.

We have also proposed using the additively homomorphic Joye-Libert cryptosystem over the more frequently used Paillier cryptosystem. Our results indicate that the drawback, in terms of computational latency of the decryption algorithm, was much smaller than anticipated. Therefore, we believe that the Joye-Libert cryptosystem would also be an appropriate cryptosystem for semi-encrypted control systems, which have previously been implemented using the Paillier cryptosystem.

Chapter 5

Towards Oblivious Guidance Systems for Autonomous Vehicles

This chapter is based on the publication

[77] P. Solnør, S. Petrovic, and T. I. Fossen, "Towards oblivious guidance systems for autonomous vehicles," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2023. (Accepted, in press)

5.1 Introduction

Several industries are adopting cloud-computing technology because of its unique benefits, including easy maintenance, distributed and remote computation, and software-as-a-service [32]. In feedback control, cloud-computing technology is central to the developing fields of cloud robotics [34, 35] and cloud control systems [12, 164], where we can host parts of the feedback control loop remotely. By enabling control-as-a-service, these systems may find use in robot swarms, intelligent transportation systems, smart grids, and process industry [140, 165, 142, 139]. However, by introducing communication links and remote computation, such schemes also bring significant concerns related to cybersecurity.

Cybersecurity of connected and autonomous vehicles is a topic undergoing intense research and covers a wide range of threats targeting sensor, operating, control, and communication systems [166]. For example, vehicles sharing information with and receiving information from other vehicles and surrounding infrastructure require privacy-preserving trust evaluation schemes [167] and reputation management systems [168] to determine whether or not to trust the information received. Broadening the scope, we must also consider expanding these trust management schemes to take advantage of vehicles and infrastructure belonging to different domains, such as aerial vehicles and satellites [169].

We can categorize cyberattacks targeting control systems as *active* or *passive*. Active attacks are, for example, data injection and spoofing. They can be used to manipulate the behavior of, or even hijack, processing plants and autonomous vehicles and, therefore, pose a significant threat. To a great extent, we can prevent

active attacks by imposing conventional cryptographic authentication protocols on the transmitted data [75] or by using anomaly detection systems [170]. On the contrary, passive attacks, for example, eavesdropping, mainly extract information from the system. Therefore, one might conclude that passive attacks are less of a threat since they do not affect the underlying physical processes. However, an attacker can leverage the information obtained from passive attacks to plan active attacks against the plant or vehicle at a later stage. An attacker can also use unauthorized eavesdropping to conduct industrial espionage, which incurs a significant risk to high-tech firms. As a result, preventing such attacks is crucial. To this end, we can use state-of-the-art stream ciphers to achieve confidential signal transmission without inducing significant time delays [74].

When considering control systems hosted on cloud infrastructure, it is insufficient to ensure confidentiality across the transmission channels since information must be decrypted and processed on third-party infrastructure, which is not necessarily trusted. Therefore, to prevent data leaks and make cloud-based control systems feasible for industrial applications, we must develop secure methods that prevent the cloud from accessing confidential system information in the first place. One way of solving this problem is by designing real-time *encrypted* feedback control systems that perform arithmetic directly on encrypted data and hence, deny unauthorized third-party providers access to unencrypted data. We can do this by using a cryptographic concept called *homomorphic encryption* [36].

Several previous studies have used homomorphic encryption to design encrypted control systems, for example, [56, 58, 72, 144, 57]. However, these studies have focused on control systems that produce an encrypted control output, which is then sent to and decrypted near an actuator. In the case of an autonomous vehicle, it would be counter-intuitive to outsource low-level control if the guidance system that produces the reference signal, for example, the desired heading or the desired course, has to run onboard the vehicle. Yet, no studies have investigated the design of encrypted guidance systems. This is precisely what we want to accomplish in this chapter; to show that we can design encrypted guidance systems that produce encrypted course or heading commands from encrypted position measurements, encrypted waypoints, and the bearing between each waypoint without inducing intolerable computational delays. We show an illustration of the concept in Fig. 5.1, where we consider two entities; the customer and the guidance provider. We assume that the communication links are secure, and the objective of the guidance provider is to determine the vehicle's position. To this end, we assume that the guidance provider is *honest-but-curious*, meaning that it will perform computations as prescribed without actively tampering with or injecting spoofed data. This assumption is motivated by the observation that reliable remote computation is key to the value proposition of cloud computing services.

5.1.1 Related works

Kogiso and Fujita [56] first built encrypted control systems using the multiplicatively homomorphic cryptosystems RSA [50] and Elgamal [51]. Since multiplicatively homomorphic cryptosystems cannot perform homomorphic addition, the proposed control systems can only compute encrypted summands, which then must be sent

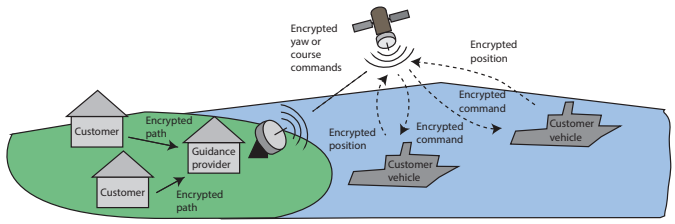


Figure 5.1 Conceptual illustration of encrypted guidance-as-a-service. The guidance provider operates on encrypted position measurements, encrypted waypoints, and the bearing between each waypoint and hence remains oblivious to the position of the customer vehicles.

back to the plant for decryption and summation. As a result, multiplicatively homomorphic cryptosystems cause an increase in data traffic. Moreover, multiplicatively homomorphic cryptosystems cause increased computational latency since the decryption algorithm must decrypt each summand. Finally, without the possibility of homomorphic additions, integral action in the control law becomes infeasible. Despite these drawbacks, multiplicatively homomorphic cryptosystems are still being used to design encrypted control systems since all the information stored in the cloud is kept encrypted [144, 145, 146].

Farokhi *et al.* [58] have proposed using additively homomorphic cryptosystems to design encrypted control systems. By viewing multiplication as repeated additions, additively homomorphic cryptosystems allow homomorphic multiplications with plaintext constants. We can use this property to build *semi-encrypted* control systems, where we store the control parameters in plaintext in the cloud, and the control systems act directly on encrypted data. As a result, researchers have used additively homomorphic cryptosystems to implement semi-encrypted linear control systems [58, 72, 71], model-predictive control [61, 59, 60, 171], and cooperative control systems [63, 62]. These studies used the Paillier cryptosystem [52] since it is readily available from numerous open-source software libraries.

Cheon *et al.* [73] later built a cryptosystem called linear homomorphic authenticated encryption with a cryptographic concept called labeled programs and used the new cryptosystem to design and implement encrypted and authenticated control. The unique benefit of the proposed system is that it includes authentication of the computational operations and the data involved. Therefore, the user detects if the cloud deviates from the planned operations or injects false data. We note that linear homomorphic authenticated encryption only allows homomorphic additions and homomorphic multiplications with plaintext constants. Barbosa *et al.* [147] used labeled programs to develop a related concept called labeled homomorphic encryption, which extends additively homomorphic cryptosystems to allow a single homomorphic multiplication. Alexandru and Pappas [65] later built an encrypted linear quadratic gaussian by showing that labeled homomorphic encryption can be used to homomorphically multiply several ciphertexts at the cost of increased data traffic. The problem with these concepts is that they require knowledge of the labels associated with the data that produced a given ciphertext for successful decryption.

For encrypted guidance, this is problematic since it requires transmitting the labels associated with each waypoint to the vehicle, at which point one might transmit the desired waypoints directly.

A significant theoretical problem concerning encrypted guidance and encrypted control is the design of encrypted stateful systems. Encrypted stateful systems are problematic because the plaintext space in which we operate usually consists of integers. Hence, we need to map our real-valued variables to integers, a mapping that blows up when we perform recursive homomorphic multiplications to update the state. Methods to avoid this problem include constraining state variables to integers [69], periodically resetting the system [70], and re-encrypting the state to remove the cumulative scaling factors [56]. Other theoretical considerations include determining appropriate key lengths, that is, security margins, in the presence of adversaries when considering the lifespan of the dynamical system [172].

Concerning implementations and proofs of concepts of encrypted control, Schulze Darup *et al.* [57] argue that few studies implement, demonstrate, and discuss practical considerations of encrypted control systems. Teranishi *et al.* [151] developed and examined an encrypted control system built using the multiplicatively homomorphic Elgamal cryptosystem. Semi-encrypted control designed using the Paillier cryptosystem was demonstrated by Farokhi *et al.* [72] and Tran *et al.* [71], where the former used a software implementation to control an indoor wheeled robot while the latter implemented their encrypted control system on a field-programmable gate array and used it to control an inverted pendulum. Cheon *et al.* [73] demonstrated their controller built using linear homomorphic authenticated encryption in a controlled laboratory experiment on an unmanned aerial vehicle.

Interestingly, even though some studies have examined encrypted vehicular control, none of the aforementioned studies have considered the design of encrypted guidance systems for path following. Intuitively, guidance systems form an 'outer' feedback loop in conventional guidance, navigation, and control systems. Therefore, a remotely hosted guidance system would seem like a more natural choice than the inner-loop controller. However, when designing encrypted guidance systems, we are faced with trigonometric functions, a unique challenge not encountered with encrypted control systems. Affine transformations, consisting of rotation matrices and translations, are used to transform coordinates from an *Earth-fixed* reference system, for example, the NED local tangent plane, to a *path-fixed* frame. In addition, the saturating arctangent function is a core component of virtually all guidance laws. Evaluating rotation matrices and the arctangent function on encrypted data is not straightforward since the algebraic structure in which we operate is a commutative ring, where the mathematical operations available consist of addition and multiplication. Hence, we cannot evaluate trigonometric functions directly. Moreover, local approximations, for example, a Taylor series, are computationally expensive to evaluate over encrypted data and are not feasible if the domain of the function is large. As a result, the problem we address in this chapter is to design encrypted guidance systems that provide adequate security. We achieve this by keeping the 'most important' information secret while also ensuring that the resulting guidance systems possess desirable stability properties and are practical from a computational point of view.

5.1.2 Main contributions

We conceptualize and investigate the design and implementation of encrypted guidance systems for straight-line path following. We argue that encrypted guidance systems are more useful than encrypted control systems for autonomous vehicles and may even be considered a prerequisite to making encrypted control systems viable for vehicular control. We propose linearized guidance laws appropriate for course and heading autopilots and show that the proposed guidance laws possess desirable stability properties. We describe how these guidance laws can be implemented in encrypted form and show that the encrypted guidance laws are computationally efficient and appropriate for real-time control. We then implement and validate an encrypted guidance system and demonstrate that it is robust to environmental disturbances through extensive field experiments using a USV in an uncontrolled environment.

5.1.3 Outline

The rest of the chapter is structured as follows. We introduce our notation and some necessary concepts from algebra, number theory, and cryptography that we use throughout this chapter in Section 5.2. Section 5.4 introduces some basic guidance concepts before we propose and analyze a set of linearized guidance laws and describe how to implement them in encrypted form in Section 5.5. We perform initial simulation tests in Section 5.6 before we present a case study in Section 5.7, where an encrypted guidance system is implemented and validated in an extensive field experiment using a USV. We discuss the obtained results, practical considerations, drawbacks, and limitations of the proposed method in Section 5.8. Finally, Section 5.9 concludes the chapter.

5.2 Notation and a brief overview of cryptographic concepts

We will denote the set of real numbers, integers, and non-negative integers by $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{Z}^+$, respectively. For $a, b \in \mathbb{Z}$, the operation $a \mid b$ reads 'a divides b', and we let $\gcd(a, b)$ denote *the greatest common divisor* of a and b. We say that a is *co-prime* to b if $\gcd(a, b) = 1$, and if $n \mid (a - b)$ for $n \in \mathbb{Z}$, we say that a is *congruent* to b modulo n, which we write as $a \equiv b \pmod{n}$. For any non-zero $n \in \mathbb{Z}^+$ and $a \in \mathbb{Z}$, $a \bmod n$ represents the smallest element in the set $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ congruent to a modulo n, and $a \bmod s$ denotes the *absolute smallest residue* of a modulo n, that is, $a \in \{\frac{-n}{2} ..., 0, ..., \frac{n-1}{2}\}$. We let $\mathbb{Z}_n^\times \coloneqq \mathbb{Z}_n \setminus \{a \in \mathbb{Z}_n \mid \gcd(a, n) \neq 1\}$ denote the multiplicative group of integers modulo n. When we use the term *plaintext*, we refer to data that is not encrypted, and we let $\mathcal{M}$ denote the *plaintext space*. The term *ciphertext* refers to an encrypted value, and we let $\mathcal{C}$ denote the *ciphertext space*. Finally, when we use the term *cryptosystem*, we refer to a full specification of a cryptographic system meant to provide *confidentiality*, including complete information about the keys, the plaintext space, the ciphertext space, the encryption algorithm, and the decryption algorithm.

Consider two groups, $G = (X, \star)$, $H = (Y, *)$, and a mapping $\phi : X \mapsto Y$. We define *homomorphism*, for example, as defined by Stinson and Paterson [152, p. 533].

Definition 6 (Group homomorphism)**.** *A homomorphism from a group $G = (X, \star)$ to a group $H = (Y, *)$ is a mapping $\phi : X \mapsto Y$ such that $\phi(a \star a') = \phi(a) * \phi(a')$, $\forall\, a, a' \in X$.*

If the encryption operation of a cryptosystem is a homomorphism $(\mathcal{M}, +) \mapsto (\mathcal{C}, *)$, where '+' denotes addition in plaintext space and '*' denotes an arbitrary group operation in ciphertext space, we call the cryptosystem *additively homomorphic*. Similarly, we refer to a cryptosystem whose encryption operation is a homomorphism $(\mathcal{M}, \cdot) \mapsto (\mathcal{C}, *)$, where '$\cdot$' denotes multiplication in plaintext space and '*' is an arbitrary group operation in ciphertext space, as *multiplicatively homomorphic*. Finally, we refer to a cryptosystem that is both additively and multiplicatively homomorphic as *fully homomorphic*.

5.2.1 Elements from number theory

For two integers $n \geq 2$ and $a \in \mathbb{Z}_n^\times$, we say that a is a *quadratic residue* modulo n if there exists an $x \in \mathbb{Z}_n^\times$ such that $x^2 \equiv a \pmod{n}$. We call a a *quadratic non-residue* if there exists no such solution. Moreover, we let Q_n denote the set of quadratic residues modulo n, and we let $\bar{Q}_n$ denote the set of quadratic non-residues modulo n. Euler's criterion now states that if p is an odd prime, an integer a is a quadratic residue modulo p if and only if $a^{\frac{p-1}{2}} \equiv 1 \pmod{p}$. We use this to define the *Legendre symbol*

$$\left(\frac{a}{p}\right) \coloneqq \begin{cases} 0, & \text{if } p|a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \bar{Q}_p. \end{cases} \tag{5.1}$$

There are several ways to generalize the concept of quadratic residues. In this chapter, we consider n^{th} *power residues*. Given two integers $n, N \geq 2$ and $a \in \mathbb{Z}_N^\times$, we call a an n^{th} power residue modulo N if there exists an $x \in \mathbb{Z}_N^\times$ such that $x^n \equiv a \pmod{N}$ and an n^{th} power non-residue if no such x exists. Along the lines of Euler's criterion, for a prime p, it holds that a is an n^{th} power residue modulo p if and only if $a^{\frac{p-1}{\gcd(n,p-1)}} \equiv 1 \pmod{p}$, and we define the symbol as $\left(\frac{a}{p}\right)_n \coloneqq a^{\frac{p-1}{n}} \bmod p$.

All integers $N \in \mathbb{Z}^+$ have a unique prime factorization, and for an odd integer $N \geq 3$ whose prime factorization is given by $N = p_1^{e_1} \cdot \ldots \cdot p_k^{e_k}$, we let

$$\left(\frac{a}{N}\right) \coloneqq \left(\frac{a}{p_1}\right)^{e_1} \cdot \ldots \cdot \left(\frac{a}{p_k}\right)^{e_k} \tag{5.2}$$

define the *Jacobi symbol*, and we let J_N denote the multiplicative group of elements whose Jacobi symbol is 1. Note that $a \in J_N \not\Rightarrow a \in Q_N$. We call such an element a *pseudosquare* modulo N, and we let $\tilde{Q}_N = J_N \setminus Q_N$ denote the set of pseudosquares modulo N. This leads us to the *quadratic residuosity problem* as defined by Menezes *et al.* [153, p. 99].

Definition 7 (Quadratic Residuosity Problem)**.** *Given an odd composite integer N and $a \in J_N$, decide whether or not $a \in Q_N$.*

For $N = pq$, where p and q are primes, it turns out that $|\tilde{Q}_N| = |Q_N|$, so a random guess has a probability of $1/2$ of being correct. Moreover, no efficient method to solve the quadratic residuosity problem is known if the factorization of N is unknown. This gives rise to the *quadratic residuosity assumption*:

Definition 8 (Quadratic Residuosity Assumption)**.** *For a random element $a \in J_N$ it is hard to determine if $a \in Q_N$ if the factorization of N is unknown.*

The security of the Goldwasser-Micali cryptosystem [154], the first *probabilistic* public-key cryptosystem, is based on the quadratic residuosity assumption. This leads to the notion of *semantic security*, for which we borrow the definition from [153, p. 306]:

Definition 9 (Semantic security)**.** *A public-key encryption scheme is said to be semantically secure if, for all probability distributions over the message space, whatever a passive adversary can compute in expected polynomial time about the plaintext given the ciphertext, he/she can also compute in expected polynomial time without the ciphertext.*

5.2.2 Joye-Libert Cryptosystem

The Goldwasser-Micali cryptosystem is not very practical since it operates on individual bits. There exist several semantically secure generalizations of the Goldwasser-Micali cryptosystem that operate on integers instead of individual bits. The Joye-Libert cryptosystem [155], which we consider in this chapter, is one of these generalizations. The security of the Joye-Libert cryptosystem is also based on the quadratic residuosity assumption and consists of a tuple, (KeyGen, Enc, Dec), defined as in [156][1] where:

- KeyGen(1^λ): Let k denote the size (in bits) of the messages being encrypted. Given a security parameter λ, KeyGen randomly generates two primes p and q, of approximately the same size in bits, such that $p \equiv 1 \pmod{2^k}$, and sets $N = pq$. It also picks $y \in \tilde{Q}_N$. The public key is then $pk = \{N, y, k\}$, and the private key is $sk = \{p\}$. The plaintext space is given by $\mathcal{M} = \mathbb{Z}_{2^k}$ and the ciphertext space is given by $\mathcal{C} = \mathbb{Z}_N^\times$.
- Enc(pk, m): To encrypt a plaintext $m \in \mathcal{M}$, pick a random $x \in \mathcal{C}$ and return $c = y^m x^{2^k} \bmod N \in \mathcal{C}$.
- Dec(sk, c): To decrypt a ciphertext $c \in \mathcal{C}$, the algorithm computes $z = (\frac{c}{p})_{2^k}$ and then finds $m \in \mathcal{M}$ such that the relation

$$z = \left[\left(\frac{y}{p}\right)_{2^k}\right]^m \bmod s\, p$$

holds.

[1] [156] contains minor corrections from the original paper by Joye and Libert [155].

As is the case for all cryptosystems, for any $m \in \mathcal{M}$, it holds that

$$\textsc{Dec}(sk, \textsc{Enc}(pk, m)) = m. \tag{5.3}$$

It is also the case that, given $m_1, m_2 \in \mathcal{M}$ such that $m_1 + m_2 \in \mathcal{M}$, it holds that

$$\textsc{Dec}(sk, \textsc{Enc}(pk, m_1) \cdot \textsc{Enc}(pk, m_2) \bmod N) = m_1 + m_2, \tag{5.4}$$

which means that the Joye-Libert cryptosystem is additively homomorphic. Moreover, we can perform homomorphic subtractions by multiplying with the multiplicative inverse of an element since $\mathcal{C}$ is a multiplicative group. Finally, given $m, k \in \mathcal{M}$ such that $km \in \mathcal{M}$, it holds that

$$\textsc{Dec}(sk, \textsc{Enc}(pk, m)^k \bmod N) = km, \tag{5.5}$$

which means that we may perform homomorphic multiplication with plaintext constants. In this chapter, we let $\boxplus, \boxminus$, and $\odot$ denote homomorphic addition, homomorphic subtraction, and homomorphic multiplication with plaintext constants, respectively. Regular $+$, $-$, and $\cdot$ denote addition, subtraction, and multiplication in $\mathcal{M}$ or $\mathcal{C}$. We denote ciphertexts in $\mathcal{C}$ with a lowercase c, whose subscript indicates the corresponding plaintext.

5.3 Joye-Libert vs Paillier

The Paillier cryptosystem [52] has the same homomorphic properties as the Joye-Libert cryptosystem and is typically used in semi-encrypted control. However, we choose to use the Joye-Libert cryptosystem for the following reasons:

1. The plaintext space of the Joye-Libert cryptosystem is $\mathbb{Z}_{2^k}$, where k is a parameter we can choose, while the plaintext space of the Paillier cryptosystem is given by $\mathbb{Z}_N^\times$, where the size of N is on the order of $2048 - 3072$ bits. We expect a $k \ll 2048$ to be sufficient.

2. The ciphertext space of the Joye-Libert cryptosystem is $\mathbb{Z}_N^\times$, while the ciphertext space of the Paillier cryptosystem is $\mathbb{Z}_{N^2}^\times$. Hence, the size of a Joye-Libert ciphertext is half the size of a Paillier ciphertext in bits. Moreover, the homomorphic operations are performed modulo N instead of modulo N^2.

3. The runtimes of the Joye-Libert encryption and decryption algorithms scale with k, while the Paillier encryption and decryption algorithms do not.

From 1) and 2), it follows that homomorphic operations are much faster with Joye-Libert plaintexts and ciphertexts than with Paillier plaintexts and ciphertexts. Moreover, 2) implies that the amount of data we transmit is reduced by 50% if we use the Joye-Libert cryptosystem. Finally, it follows from 3) that we can expect the Joye-Libert encryption and decryption algorithms to be fast for small k.

5.4 The line-of-sight guidance principle

We consider underactuated vehicles with a 3-degrees-of-freedom maneuvering model of the form [81, p. 157]

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}_b^n(\psi)\boldsymbol{\nu} \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{N}(\boldsymbol{\nu})\boldsymbol{\nu} &= \begin{bmatrix} \tau_1 \\ 0 \\ \tau_3 \end{bmatrix}, \end{aligned} \tag{5.6}$$

where $\boldsymbol{\eta} = [x^n, y^n, \psi]^T \in \mathbb{R}^2 \times \mathbb{S}$ describes the vehicle pose in the Earth-fixed NED reference frame, $\boldsymbol{\nu} = [u, v, r]^T \in \mathbb{R}^3$ describes the vehicle velocity in the body-fixed frame, $\boldsymbol{R}_b^n(\psi) \in \mathrm{SO}(3)$ is a rotation matrix from the body-fixed frame to the NED frame, ψ is the yaw angle, $\boldsymbol{M}$ is the mass-inertial matrix, and $\boldsymbol{N}(\boldsymbol{\nu})$ describes the Coriolis, centripetal, and damping forces. The speed of the vehicle is $U = \sqrt{u^2 + v^2}$.

We now consider straight-line path-following between two successive waypoints. We use the NED frame as the reference frame, and we use a path-fixed frame whose origin we fix to the first waypoint and whose x-axis is tangential to the reference path to describe the position of the vehicle relative to the reference path. We write the coordinates of a 2-D point (x, y) with respect to the NED frame as (x^n, y^n) and with respect to the path-fixed frame as (x^p, y^p). A rotation matrix

$$\boldsymbol{R}_p^n(\pi_p) = \begin{bmatrix} \cos(\pi_p) & -\sin(\pi_p) \\ \sin(\pi_p) & \cos(\pi_p) \end{bmatrix} \in \mathrm{SO}(2) \tag{5.7}$$

and a translation from the NED origin to the first waypoint relate coordinates in the NED frame to coordinates in the path-fixed frame, where the angle π_p between two waypoints (x_i^n, y_i^n) and (x_{i+1}^n, y_{i+1}^n) is given by

$$\pi_p = \mathrm{atan2}\left(y_{i+1}^n - y_i^n, x_{i+1}^n - x_i^n\right). \tag{5.8}$$

We can now compute the cross-track error according to

$$\begin{bmatrix} 0 \\ y_e^p \end{bmatrix} = \boldsymbol{R}_p^n(\pi_p)^T \begin{bmatrix} x^n - x_i^n \\ y^n - y_i^n \end{bmatrix}, \tag{5.9}$$

where the along-track error is defined as zero. The lookahead-based LOS guidance law is given by

$$\chi_d = \pi_p - \tan^{-1}\left(\frac{y_e^p}{\Delta}\right), \tag{5.10}$$

where the output $\chi_d \in [-\pi, \pi)$ is the *desired course* expressed in the NED frame, $\Delta > 0$ is the *look-ahead distance*, and $1/\Delta$ is interpreted as the proportional gain. It can be shown, using Lyapunov stability theory, that the proportional LOS guidance law is uniformly semi-globally exponentially stable [173].

In practice, marine vehicles are equipped with a yaw autopilot, which used in conjunction with (5.10) does not guarantee convergence to the path because of environmental disturbances, such as winds, waves, and ocean currents. In such cases, the course and the heading of the vehicle differ by an angle β_c, called the *crab*

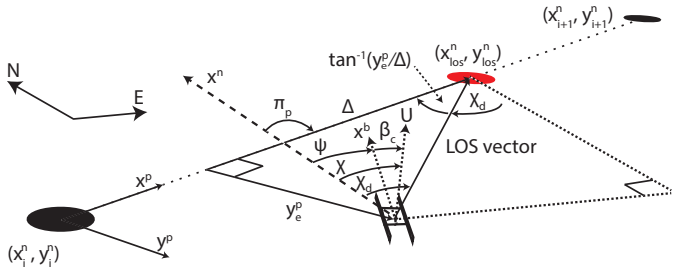


Figure 5.2 Overview of the coordinate frames used and the lookahead-based LOS guidance principle.

angle, such that $\chi = \psi + \beta_c$. Assuming β_c is slowly varying, we can add integral action to compensate for this offset by allowing the vehicle to sideslip. We refer to a LOS guidance law with integral action as an ILOS guidance law. A frequently used ILOS guidance law is

$$\psi_d = \pi_p - \tan^{-1}\left(k_p y_e^p + k_i \int_0^t y_e^p \, d\tau\right), \tag{5.11}$$

where $k_p > 0$, $k_i > 0$, the proportional and the integral gains, respectively, are design parameters, and where the output, $\psi_d \in [-\pi, \pi)$, is the *desired heading* of the vehicle. We can show that the above ILOS guidance law is locally stable by assuming $U > 0$ is constant and by using linearization about $y_e^p = 0$, but there are no results for global stability. Moreover, choosing appropriate gains for (5.11) is challenging because of problems with integral windup [81, p. 364]. Figure 5.2 summarizes and shows an overview of the relevant angles and coordinate frames.

5.5 Design of encrypted guidance systems using additively homomorphic encryption

Remotely hosting proportional LOS and ILOS guidance algorithms on cloud infrastructure is straightforward, and the communication can be secured using symmetric cryptography without inducing significant computational delays [74]. However, such a design requires the waypoints to be stored in unencrypted form on the cloud infrastructure. Moreover, the position measurements from the vehicle must be decrypted in the cloud, enabling real-time monitoring of the vehicle's position. For certain customers, such leaks of information are highly discouraging. Therefore, we wish to adapt the guidance algorithms such that we can evaluate the guidance laws on encrypted data, avoiding the need to store unencrypted waypoints and position measurements in the cloud. To achieve this, we intentionally reveal the bearing π_p between each waypoint. The idea is that leaking the general direction in which a vehicle should move significantly simplifies the computation. Moreover, it does not reveal any information about the vehicle's position, provided the position measurements and the waypoints are encrypted. The reasoning is that the general direction in which a vehicle should move is insufficient information to determine

the vehicle's absolute position if all positioning information remains encrypted. Moreover, relative positioning is infeasible without access to the velocity. To obtain this information, the guidance provider must decrypt the position measurements or the waypoints without access to the decryption key. Hence, the scheme's security follows from the security of the underlying cryptosystem.

5.5.1 Linearization of LOS guidance laws

We start by considering the lookahead-based LOS guidance law (5.10) appropriate for vehicles with course autopilots. Since trigonometric functions are not feasible to evaluate homomorphically with an additively homomorphic cryptosystem, we use the observation that $\tan^{-1}(x)$ is just a saturated linear function and define our proposed linearized LOS guidance law as

$$\chi_d \coloneqq \pi_p - k_p y_e^p, \tag{5.12}$$

where $k_p > 0$, the proportional gain, is a design parameter. We consider the stability properties of (5.12) using the following Lyapunov candidate function

$$V(y_e^p) = \frac{1}{2}(y_e^p)^2, \tag{5.13}$$

where $V(y_e^p) > 0$ whenever $y_e^p \neq 0$. The cross-track error dynamics is given by

$$\dot{y}_e^p = U \sin(\chi - \pi_p). \tag{5.14}$$

We then assume a non-zero, but possibly time-varying, speed,[2] that is, $U \geq U_{\min} > 0$, and perfect course control such that $\chi = \chi_d$. Differentiating (5.13) with respect to time and inserting (5.14) then yields

$$\begin{aligned}
\dot{V} &= y_e^p \dot{y}_e^p \\
&= y_e^p U \sin(\chi - \pi_p) \\
&= y_e^p U \sin(-k_p y_e^p) \\
&\leq -y_e^p U_{\min} \sin(k_p y_e^p) \\
&< 0 \; \forall \; y_e^p \neq 0 \in \left(-\frac{\pi}{k_p}, \frac{\pi}{k_p} \right).
\end{aligned} \tag{5.15}$$

Since $V(y_e^p) > 0$ and $\dot{V}(y_e^p) < 0 \; \forall \; y_e^p \neq 0$, we have that $|y_e^p(t)| \leq |y_e^p(t_0)| \; \forall \; t > t_0$ and $|y_e^p(t)| = |y_e^p(t_0)| \iff y_e^p(t_0) = 0 \; \forall \; t > t_0$. Hence, we conclude that the origin $y_e^p = 0$ is uniformly asymptotically stable for $y_e^p(t_0) \in (-\frac{\pi}{k_p}, \frac{\pi}{k_p})$. Moreover, using $\sin(x) \approx x$ for $|x| \ll 1$, we have

$$\begin{aligned}
\dot{V} &= -y_e^p U \sin(k_p y_e^p) \\
&\approx -U k_p (y_e^p)^2 \\
&\leq -2 U_{\min} k_p V,
\end{aligned} \tag{5.16}$$

[2] We have to assume a non-zero speed since the course of a vehicle is not defined for U = 0. Typically we would also assume a time-varying Δ, but this is impractical in encrypted form.

from which we conclude that the guidance law is exponentially stable for $|y_e^p| \ll 1/k_p$, according to [174, Th. 4.10].

If we instead consider vehicles with yaw autopilots, we have to compensate for the disturbance β_c. Assuming the waves, ocean current, and wind cause a slowly varying β_c, we add integral action and consider the following linearization of (5.11)

$$\psi_d \coloneqq \pi_p - k_p y_e^p - k_i \int_0^t y_e^p \, d\tau. \tag{5.17}$$

We can rewrite (5.17) as

$$\psi_d = \pi_p - k_p y_e^p - k_i y_{\text{int}}^p \tag{5.18}$$
$$\dot{y}_{\text{int}}^p = y_e^p. \tag{5.19}$$

To study the stability properties of (5.17), we note that for $\chi \approx \pi_p$, we have

$$\begin{aligned} \dot{y}_e^p &= U \sin(\chi - \pi_p) \\ &\approx U(\chi - \pi_p) \\ &= U(\psi + \beta_c - \pi_p). \end{aligned} \tag{5.20}$$

Assuming perfect heading control, such that $\psi = \psi_d$, and by inserting (5.18) into (5.20), we get the following linearized system

$$\dot{y}_e^p = U(-k_p y_e^p - k_i y_{\text{int}}^p + \beta_c) \tag{5.21}$$
$$\dot{y}_{\text{int}}^p = y_e^p, \tag{5.22}$$

from which we find that the system has an equilibrium point $(y_e^p, y_{\text{int}}^p) = (0, \beta_c/k_i)$. We set $x_1 = y_e^p$ and $x_2 = y_{\text{int}}^p - \beta_c/k_i$ and consider the system

$$\dot{x}_1 = -U k_p x_1 - U k_i x_2 \tag{5.23}$$
$$\dot{x}_2 = x_1, \tag{5.24}$$

which is of form $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x}$. Assuming $U > 0$ is constant, $\boldsymbol{A}$ is Hurwitz, and we conclude that (5.23)–(5.24) is asymptotically stable and that (5.17) is locally exponentially stable.

While we do not attain global stability with the proposed linearizations, we point out that from a practical point of view, local stability is often sufficient since vehicles tend to stay 'close' to their desired paths. Moreover, we note that exponential stability indicates that the proposed guidance laws are robust against nonvanishing uniformly bounded perturbations, such as wind gusts and waves [174, Lemma 9.2]. Indeed, the most significant corollary of losing global stability is that we must initialize a vehicle using a locally stable guidance law within its stability region.

5.5.2 From real numbers to valid plaintexts

To implement (5.12) and (5.17), we have to map the position of the vehicle, the guidance parameters, and the waypoints from some real interval $\mathcal{I} = [a, b]$, usually

represented as floating-point numbers, to the plaintext space $\mathcal{M}$ of the Joye-Libert cryptosystem. Since $\mathcal{M}$ consists of the commutative ring of integers $\mathbb{Z}_{2^k} = \{0, 1, \ldots, 2^k - 1\}$, we adopt the map $\rho : \mathcal{I} \times \mathbb{Z}_{2^k} \mapsto \mathbb{Z}_{2^k}$ defined as

$$\rho(x, \gamma) \coloneqq \begin{cases} \lceil \gamma x \rfloor, & \text{if } x \geq 0 \\ \lceil \gamma x \rfloor + 2^k, & \text{otherwise,} \end{cases} \tag{5.25}$$

where γ is a scaling constant and $\lceil \cdot \rfloor$ denotes rounding to the nearest integer. Given $x \in \mathcal{I}$, we let $\bar{x}$ denote the corresponding plaintext output of $\rho(x, \gamma)$. Clearly, (5.25) induces a quantization error

$$\epsilon \leq \frac{b-a}{2^{k+1}}, \tag{5.26}$$

assuming all values of the plaintext space are used. Moreover, we need to consider the resulting cumulative scaling when we multiply ciphertexts with plaintext constants, given by

$$\bar{m} = \prod_{i=1}^{l} \bar{m}_i = \prod_{i=1}^{l} \gamma_i x_i, \tag{5.27}$$

where

$$\gamma = \prod_{i=1}^{l} \gamma_i \leq \frac{2^k}{b-a} \tag{5.28}$$

must hold. Fortunately, since (5.17) is of a special form where the stateful component is just represented as a cumulative sum, we will not encounter problems with a growing cumulative scaling in the state of the encrypted guidance system with integral action. Note that (5.27) also implies that the designer of the encrypted guidance system can choose increased precision of individual variables at the cost of reduced precision of other variables.

Finally, once we have performed the arithmetic in ciphertext space and decrypted the resulting ciphertext c_m, we must map the recovered plaintext $\bar{m} \in \mathcal{M}$ back to the appropriate real value $m \in \mathcal{I}$ using the following map

$$\rho(\bar{m}, \gamma)^{-1} \coloneqq \begin{cases} \frac{\bar{m} - 2^k}{\gamma}, & \text{if } \bar{m} \geq 2^{k-1} \\ \frac{\bar{m}}{\gamma}, & \text{otherwise,} \end{cases} \tag{5.29}$$

where γ is given by (5.28). According to our discussion above, this means that the cumulative scaling of each summand of $\bar{m}$ must be the same.

5.5.3 Path following using an encrypted guidance system

We are now ready to describe our encrypted guidance system. Consider a straight path from a waypoint (x_i^n, y_i^n) to a waypoint (x_{i+1}^n, y_{i+1}^n). The encrypted guidance system holds the corresponding ciphertexts $c_{p_i^n} = (c_{x_i^n}, c_{y_i^n})$ and $c_{p_{i+1}^n} = (c_{x_{i+1}^n}, c_{y_{i+1}^n})$, along with plaintexts $\overline{\sin(\pi_{p_i})}$, $\overline{\cos(\pi_{p_i})}$, and ciphertext $c_{\pi_{p_i}}$. The vehicle then quantizes, encrypts, and transmits its position (c_{x^n}, c_{y^n}) to the guidance

system, which computes the encrypted cross-track error according to

$$\begin{bmatrix} c_{x_e^p} \\ c_{y_e^p} \end{bmatrix} = \overline{\boldsymbol{R}_p^n(\pi_p)}^T \begin{bmatrix} c_{x^n} \boxminus c_{x_p^n} \\ c_{y^n} \boxminus c_{y_p^n} \end{bmatrix} \tag{5.30}$$

$$= \begin{bmatrix} (c_{x^n} \boxminus c_{x_p^n}) \odot \overline{\cos(\pi_p)} \boxplus (c_{y^n} \boxminus c_{y_p^n}) \odot \overline{\sin(\pi_p)} \\ (c_{y^n} \boxminus c_{y_p^n}) \odot \overline{\cos(\pi_p)} \boxminus (c_{x^n} \boxminus c_{x_p^n}) \odot \overline{\sin(\pi_p)} \end{bmatrix}. \tag{5.31}$$

The cloud can then compute the encrypted desired yaw c_{ψ_d} according to the encrypted guidance law

$$c_{\psi_d} = c_{\pi_p} \boxminus \bar{k}_p \odot c_{y_e^p} \boxminus \bar{k}_i \odot \boxplus_{k=1}^N c_{y_e^p}[k] \odot \bar{\Delta} t, \tag{5.32}$$

where $\boxplus_{k=1}^N$ denotes a homomorphic summation over N elements. Of course, a vehicle equipped with course control can drop the summation term if we neglect modeling errors and kinematic couplings. Moreover, since the encrypted guidance system only has access to encrypted position measurements and waypoints, it cannot assess the distance between the vehicle and the next waypoint. Therefore, the encrypted guidance system also computes the encrypted distance to the next waypoint, in the path-fixed frame, according to

$$\begin{bmatrix} c_{\tilde{x}^p} \\ c_{\tilde{y}^p} \end{bmatrix} = \overline{\boldsymbol{R}_p^n(\pi_p)}^T \begin{bmatrix} c_{x_{p+1}^n} \boxminus c_{x^n} \\ c_{y_{p+1}^n} \boxminus c_{y^n} \end{bmatrix}. \tag{5.33}$$

The guidance system then transmits the tuple $(c_{\psi_d}, c_{\tilde{x}^p}, c_{\tilde{y}^p})$ to the vehicle, which decrypts the elements, recovers $(\psi_d, \tilde{x}^p, \tilde{y}^p)$, and computes appropriate thrust allocations using a yaw controller with ψ_d as the desired yaw. The vehicle also computes

$$\delta = \max\{|\tilde{x}^p|, |\tilde{y}^p|\} \tag{5.34}$$

and notifies the encrypted guidance system when $\delta \leq \tau$, where τ is the threshold for acceptance, that is, a threshold determining when a waypoint is considered reached. We show pseudocode describing the offline preprocessing, encryption, encrypted guidance, and decryption algorithms in Algorithms 12, 13, 14, and 15, respectively. In the context of Fig. 5.1, we note that Algorithm 12 runs in the customer offices, Algorithms 13 and 15 run onboard the customer vehicles, and Algorithm 14 runs on the third-party cloud infrastructure hosting the guidance algorithm.

5.5.4 Choosing the plaintext size and the security margin

In addition to choosing the conventional guidance parameters k_p and k_i, we must also choose the number of bits used to represent each plaintext, k, and the size (in bits) of the factoring modulus N. By choosing a large k, we reduce the quantization error induced by (5.25). However, a large k also increases the computational latency induced, particularly with respect to the decryption algorithm since it recovers each bit individually. Because we are implementing the algorithms in software, the most natural choices are $k = 32$ or $k = 64$, such that we can represent each plaintext by 4-byte or 8-byte unsigned integers, respectively. Concerning the choice of N, we note that the size of N is a tradeoff between the computational latency and

Algorithm 12 Offline preprocessing

Parameters

pk	Joye-Libert public key
$\gamma_p, \gamma_\pi, \gamma_{\text{trig}}$	Scaling factors

Input

$p_1^n, \ldots, p_k^n$	Waypoints $p_i^n = (x_i^n, y_i^n)$

Output

$c_{p_1^n}, \ldots, c_{p_k^n}$	Encrypted waypoints $c_{p_i^n} = (c_{x_i^n}, c_{y_i^n})$
$c_{\pi_{p_1}}, \ldots, c_{\pi_{p_{k-1}}}$	Encrypted bearing between waypoints
$c_{\text{integral},0}$	Encrypted initial integral state
$\overline{\sin(\pi_i)}$	Plaintext values for $i \in \{1, \ldots, k-1\}$
$\overline{\cos(\pi_i)}$	Plaintext values for $i \in \{1, \ldots, k-1\}$

function OfflinePreprocessing($p_1^n, \ldots, p_k^n$)
  for $i = 1$ to $k-1$ **do**
    $\bar{p}_i^n \leftarrow \rho(p_i^n, \gamma_p)$
    $\pi_{p_i} \leftarrow \text{atan2}(y_{i+1}^n - y_i^n, x_{i+1}^n - x_i^n)$
    $\bar{\pi}_{p_i} \leftarrow \rho(\pi_{p_i}, \gamma_\pi)$
    $\overline{\sin(\pi_{p_i})} \leftarrow \rho(\sin(\pi_{p_i}), \gamma_{\text{trig}})$
    $\overline{\cos(\pi_{p_i})} \leftarrow \rho(\cos(\pi_{p_i}), \gamma_{\text{trig}})$
    $c_{p_i^n} \leftarrow \text{Enc}(pk, \bar{p}_i^n)$
    $c_{\pi_{p_i}} \leftarrow \text{Enc}(pk, \bar{\pi}_{p_i})$
  end for
  $\bar{p}_k^n \leftarrow \rho(p_k^n, \gamma_p)$
  $c_{p_k^n} \leftarrow \text{Enc}(pk, \bar{p}_k^n)$
  $c_{\text{integral, 0}} \leftarrow \text{Enc}(pk, 0)$
  Send $[c_{p_1^n}, \ldots, c_{p_k^n}, c_{\pi_{p_1}}, \ldots, c_{\pi_{p_{k-1}}}, c_{\text{integral, 0}}, \overline{\sin(\pi_1)}, \ldots, \overline{\sin(\pi_{k-1})}, \overline{\cos(\pi_1)}, \ldots, \overline{\cos(\pi_{k-1})}]$ to the guidance system.
end function

Algorithm 13 Onboard encryption

Parameters

pk	Joye-Libert public key
$\gamma_{x^n}, \gamma_{y^n}$	Scaling factors

Input

(x^n, y^n)	Position in NED frame

Output

(c_{x^n}, c_{y^n})	Encrypted position in NED frame

1: **function** QuantizeAndEncrypt(x^n, y^n)
2: $\bar{x}^n \leftarrow \rho(x^n, \gamma_{x^n})$
3: $\bar{y}^n \leftarrow \rho(y^n, \gamma_{y^n})$
4: $c_{x^n} \leftarrow \text{Enc}(\text{pk}, \bar{x}^n)$
5: $c_{y^n} \leftarrow \text{Enc}(\text{pk}, \bar{y}^n)$
6: Send (c_{x^n}, c_{y^n}) to the guidance system.
7: **end function**

the level of security. Relevant choices for N are $\log_2 N \approx 2048$ and $\log_2 N \approx 3072$, corresponding to approximately 112-bit and 128-bit security against brute-force attacks, respectively [160, Table 2].

5.5.5 Induced computational latency

For the proposed scheme to be *practical*, we must ensure that the computational latency induced by Algorithms 13, 14, and 15 is suitable for real-time operations. The computational latency mainly affects the performance of the guidance law in terms of the cross-track error by limiting the frequency with which we can iterate the guidance loop. The maximum frequency of the guidance loop is upper-bounded by $1/\mathrm{T}_{\max}$, where $\mathrm{T}_{\max}$ denotes the maximum of the computational latencies induced by Algorithms 13, 14, and 15. The effect of limiting the frequency of the guidance loop depends on the vehicle dynamics; specifically, vehicles with fast dynamics are more adversely affected. In addition, there are other mechanisms with which the computational latency negatively affects the guidance performance. For example, the computational latency reduces the phase margin of the closed-loop system proportional to the total latency induced.

We implemented the encrypted guidance system in C++ and used an implementation of the Joye-Libert cryptosystem from the CryptoToolbox, a publicly available cryptographic code repository [79]. We also implemented an alternative variant where we instantiate the algorithms with the Paillier cryptosystem to substantiate the claimed benefits of using the Joye-Libert cryptosystem. We used the GNU Multiple Precision Arithmetic Library to represent big numbers and for big-number arithmetic [162]. We then measured the computational latencies on an Nvidia Jetson Xavier with the system specifications shown in Table 5.1. Figure 5.3 shows the computational latencies induced by Algorithms 13, 14, and 15 when instantiated with the Joye-Libert and the Paillier cryptosystems. As expected, the instantiations with the Joye-Libert cryptosystem significantly outperform the instantiations with the Paillier cryptosystem. Moreover, the computational latencies induced by the instan-

Algorithm 14 Encrypted guidance system with integral action

Parameters

$\bar{k}_\mathrm{p}$	Proportional gain mapped to plaintext space
$\bar{k}_\mathrm{i}$	Integral gain mapped to plaintext space
$\bar{\Delta}t$	Time step mapped to plaintext space
$c_{\text{integral, 0}}$	Initial integral state
$c_{p_1^n}, \ldots, c_{p_k^n}$	Encrypted waypoints $c_{p_i^n} = (c_{x_i^n}, c_{y_i^n})$
$c_{\pi_{p_1}}, \ldots, c_{\pi_{p_{k-1}}}$	Encrypted bearing between waypoints
$\overline{\sin(\pi_i)}$	Plaintext values for $i \in \{1, \ldots, k-1\}$
$\overline{\cos(\pi_i)}$	Plaintext values for $i \in \{1, \ldots, k-1\}$

Input

c_{x^n}, c_{y^n}	Encrypted position in NED frame
b	Bit to indicate waypoint reached

Output

c_{ψ_d}	Encrypted yaw reference
$c_{\tilde{p}^p} = (c_{\tilde{x}^p}, c_{\tilde{y}^p})$	Encrypted position error in path-fixed frame

function EncryptedCloudGuidance
  $c_\text{integral} \leftarrow c_{\text{integral, 0}}$
  $i \leftarrow 2$
  while destination not reached **do**
    for each new message $[c_{x^n}, c_{y^n}, b]$ **do**
      $i = i + b$
      if $i > k$ **then**
        return
      end if
      $c_{y_e^p} = (c_{y^n} \boxminus c_{y_{p_i}^n}) \odot \overline{\cos(\pi_{p_i})} \boxminus (c_{x^n} \boxminus c_{x_{p_i}^n}) \odot \overline{\sin(\pi_{p_i})}$
      $c_{\psi_d} = c_{\pi_{p-1}} \boxplus \bar{k}_\mathrm{p} \odot c_{y_e^p} \boxplus \bar{k}_\mathrm{i} \odot \boxplus_{k=1}^{N} c_{y_e^p}[k] \odot \bar{\Delta}t$
      $c_{\tilde{x}^p} = (c_{x^n} \boxminus c_{x_{p+1}^n}) \odot \overline{\cos(\pi_{p_i})} \boxplus (c_{y^n} \boxminus c_{y_{p+1}^n}) \odot \overline{\sin(\pi_{p_i})}$
      $c_{\tilde{y}^p} = (c_{y^n} \boxminus c_{y_{p+1}^n}) \odot \overline{\cos(\pi_{p_i})} \boxminus (c_{x^n} \boxminus c_{x_{p+1}^n}) \odot \overline{\sin(\pi_{p_i})}$
      Send $[c_{\psi_d}, c_{\tilde{x}^p}, c_{\tilde{y}^p}]$ to the vehicle.
    end for
  end while
end function

Algorithm 15 Recovery of encrypted yaw reference

Parameters

sk	Joye-Libert secret key
γ	Cumulative scaling factor
τ	Acceptance threshold

Input

c_{ψ_d}	Encrypted desired yaw
$c_{\tilde{x}^p}$	Encrypted along-track distance
$c_{\tilde{y}^p}$	Encrypted cross-track distance

Output

ψ_d	Desired yaw
b	Waypoint reached indicator

function DecryptAndRecover($c_{\psi_d}, c_{\tilde{x}^p}, c_{\tilde{y}^p}$)
  $b \leftarrow 0$
  $\bar{\psi}_d \leftarrow \text{Dec}(c_{\psi_d}, sk)$
  $\bar{x}^p \leftarrow \text{Dec}(c_{\tilde{x}^p}, sk)$
  $\bar{y}^p \leftarrow \text{Dec}(c_{\tilde{y}^p}, sk)$
  $\psi_d \leftarrow \rho^{-1}(\bar{\psi}_d, \gamma)$
  $\tilde{x}^p \leftarrow \rho^{-1}(\bar{x}^p, \gamma)$
  $\tilde{y}^p \leftarrow \rho^{-1}(\bar{y}^p, \gamma)$
  $\delta \leftarrow \max\{|\tilde{x}^p|, |\tilde{y}^p|\}$
  if $\delta \leq \tau$ **then**
    $b \leftarrow 1$
  end if
  Send b to the guidance system.
  Send ψ_d to the control system.
end function

tiations with the Joye-Libert cryptosystem scale with k, while the computational latencies induced by the Paillier instantiations do not. In conclusion, an encrypted guidance system instantiated with the Joye-Libert cryptosystem can operate at significantly higher frequencies than an encrypted guidance system instantiated with the Paillier cryptosystem. It follows that we expect an encrypted guidance system instantiated with the Joye-Libert cryptosystem to result in better guidance performance in terms of the cross-track error, particularly for vehicles with fast dynamics.

5.6 Simulation results

We begin by validating the proposed encrypted guidance system through simulations on a first-order Nomoto model for heading control [81, p. 188] given by

$$\ddot{\psi} = -\frac{1}{T}\dot{\psi} + \frac{K}{T}\tau_3, \tag{5.35}$$

where τ_3 is the heading control input from (5.6), K is the Nomoto gain constant, and T is the Nomoto time constant. To demonstrate our encrypted guidance system,

Table 5.1 System specifications for the experiments

Hardware	
Model name	NVIDIA Jetson Xavier
CPU	NVIDIA Tegra Xavier
Instruction set architecture	ARMv8.2
Number of cores	8
Word size	64 bit
Memory	32GB
Software	
Operating system	Ubuntu 18.04 LTS
Big number library	GMP 6.2.1
Compiler	g++ 7.5.0

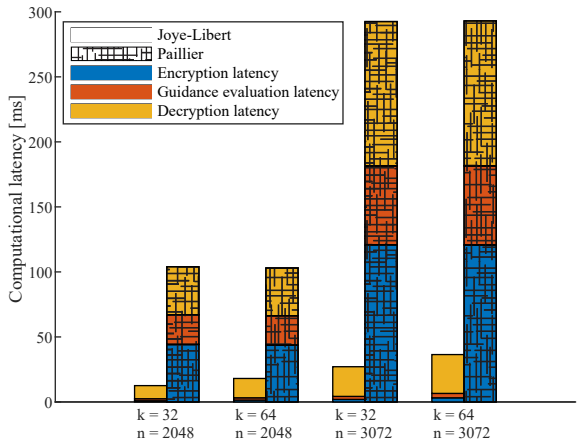


Figure 5.3 Computational latencies induced by the encryption, encrypted guidance, and decryption algorithms for k-bit plaintexts and an n-bit factoring modulus instantiated with the Joye-Libert and Paillier cryptosystems.

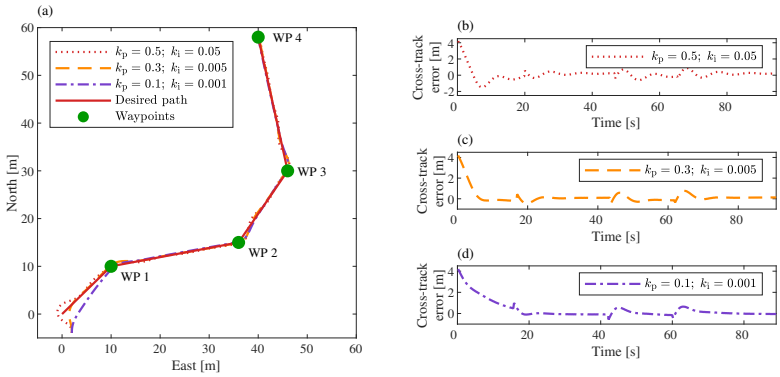


Figure 5.4 Simulations of the proposed encrypted guidance system with a first-order Nomoto model for heading control, using a range of guidance law gains. In (a) we show the simulated paths against the desired path, while in (b)–(d) we show the cross-track errors for the different guidance law gains.

we set $K = 1/2$ [s^{-1}] and $T = 1$ [s]. We use a proportional heading controller $\tau_3 = k_{p_\psi}(\psi_d - \psi)$, where we set $k_{p_\psi} = 1.5$, and we fix the vehicle's speed to $U = 1$ [m/s], where we neglect the sway motion by assuming $v \approx 0$. Note that integral action is dropped in the inner control loop to avoid two controllers with integral action in cascade, which negatively affects the phase margin of the closed-loop system. We set the threshold of acceptance to $\tau = 1$ [m] and the guidance loop to run at a frequency of 5 [Hz]. The inner heading control loop runs at a frequency of 25 [Hz].

We test three parameter configurations where we initialize the vehicle with a cross-track error of 4.5 [m] and with $\psi = 0$ [rad] and $\dot{\psi} = 0$ [rad/s]. In the first simulation, we use a set of gains for our encrypted guidance system where the initial point of the USV is close to the border of stability. In the second and third simulations, we reduce the gains for the encrypted guidance system, which increases the region of attraction at the cost of reducing the convergence rate to the desired path. Figure 5.4 shows the obtained results, demonstrating that the proposed encrypted guidance system successfully guides a vehicle along a path consisting of straight-line segments. Interested readers can find the implementations of Algorithms 12, 13, 14, and 15 and the code to reproduce the simulations online [175].

5.7 Validation of an encrypted guidance system through field experiments

To assess the practical nature of the proposed encrypted guidance systems, we ported an encrypted guidance system with integral action to the Cyberotter USV, shown in Fig. 5.5. The Cyberotter is actuated by two fixed thrusters mounted at the stern on the port and starboard hulls, respectively. As a result, the Cyberotter is underactuated and can only directly control the yaw angle and the surge speed.

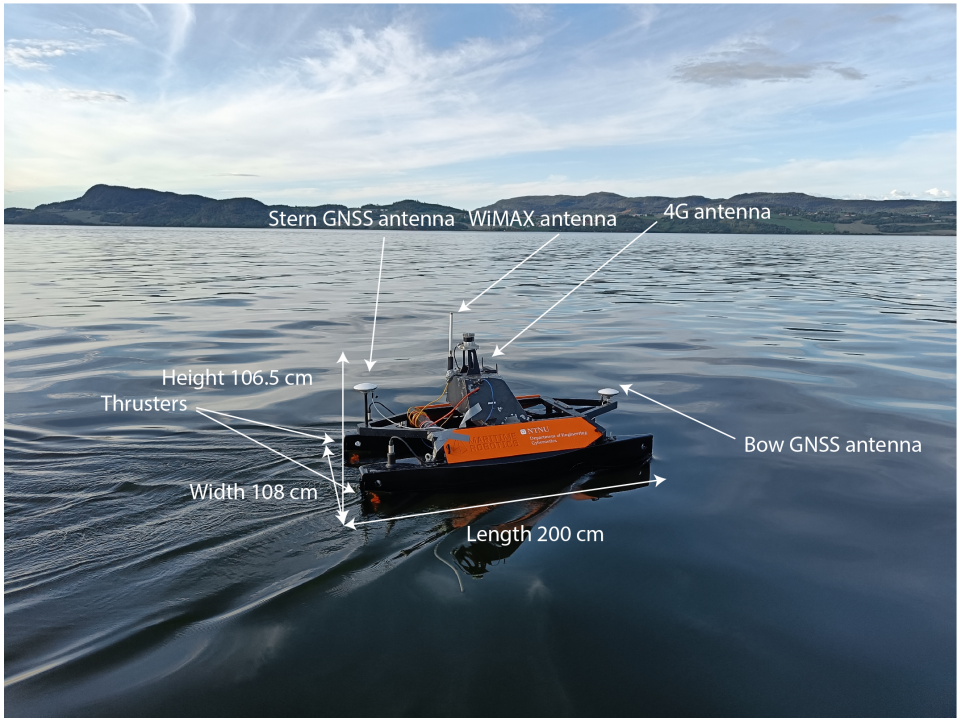


Figure 5.5 Overview of the Cyberotter USV during one of the encrypted guidance experiments in Børsa, Norway.

5.7.1 Experimental setup

We conducted the experiments in the bay area outside of Børsa, a town by the Trondheim fjord approximately 30 minutes southwest of Trondheim by car. The Trondheim fjord is a large fjord known for strong currents and significant commercial and recreational activity. As a result, the vehicle is also subject to waves and swells caused by other vessels during the experiments. For each experiment, the objective of the USV is to follow a path with straight-line segments between waypoints while maintaining a fixed surge speed. We uploaded the encrypted waypoints to the machine hosting the encrypted guidance system, an Nvidia Jetson Xavier hosted in an office building in Trondheim, along with the respective rotation matrices from the NED frame to the path-fixed frame, mapped to plaintext space. Using two industrial 4G routers, we established an IPsec tunnel between the Cyberotter and the office through which we transmitted all data between the Cyberotter and the encrypted guidance system.

The onboard motion control system consists of a proportional-integral surge speed controller and a proportional yaw controller. Since the Cyberotter is port-starboard symmetric, the surge speed controller computes a common thrust for the two thrusters, while the yaw controller produces a differential thrust. The differential thrust is then added and subtracted to the common thrust to produce the starboard

Table 5.2 Guidance parameters used during each experiment

Experiment	Path	k_{p}	k_{i}	u_d [m/s]
1	1	0.1	0.000785	1.0
2	1	0.1	0.000157	1.0
3	2	0.1	0.000785	1.0
4	2	0.1	0.000157	2.0

and port thrust allocations, respectively. When the error between the desired and the measured yaw exceeds 30 degrees, the vehicle disables the surge speed controller until it has sufficiently corrected its yaw. We set the guidance system to operate at a frequency of 3 Hz while the inner-loop control system operates at 25 Hz. The onboard navigation system consists of a commercial off-the-shelf INS with a dual-antenna GNSS receiver for yaw estimation. We used a graphical user interface hosted on a laptop to execute missions and monitor the overall performance during each experiment. Finally, we used a WiMAX connection between the vehicle and the laptop to pass data between the graphical user interface and the USV. To monitor and record the experiments, we used a recreational boat.

An Nvidia Jetson Xavier is used to encrypt the position measurements and decrypt the desired heading and the distance to the next waypoint onboard the vehicle. Since the machines involved have a 64-bit word size, we proceed by setting $k = 64$. Setting $\log_2 N \approx 3072$ results in approximately 128-bit security, which provides a comfortable level of security for the foreseeable future. Table 5.2 shows the guidance parameters used in each experiment. We choose scaling factors such that the cumulative scaling of each summand is $\gamma = 2^{61}$, meaning that we can represent numbers in the interval $[0, 2^3) = [0, 8)$, which we can shift to $\mathcal{I} = [-4, 4)$, which is sufficient for all $\psi_d \in [-\pi, \pi)$ without encountering problems with overflow or underflow. Table 5.3 shows the constants we used to map each variable to valid plaintexts and bounds on the induced rounding errors. We note that the scaling factors were chosen on an ad-hoc basis and are not an 'optimal' set of scaling factors. Regardless, it is clear from Table 5.3 that any performance degradation caused by quantization is negligible. Finally, we set the threshold of acceptance to five meters, that is, $\tau = 5$ [m].

We tested the encrypted guidance system using two distinct paths. The first path consists of six waypoints forming a circular path in the outer region of the bay area. The second path consists of five waypoints starting in the inner part of the bay, close to a river outlet which we expect to produce more varying currents and hence, more significant cross-track errors.

5.7.2 Experimental results

For each experiment, we present the results by plotting the path of the USV against the desired waypoints. Moreover, we plot the surge speed against the desired surge speed and the yaw against the desired yaw of the vehicle. Finally, we plot the cross-track error of the vehicle. We show the results from experiments 1 and 2,

Table 5.3 Scaling constants used to map variables to valid plaintexts

Scaling factor	Value	Quantization error
γ_{k_p}	2^{21}	$\lvert\epsilon_{k_p}\rvert < 2.38 \times 10^{-7}$
γ_{k_i}	2^{16}	$\lvert\epsilon_{k_i}\rvert < 7.63 \times 10^{-6}$
γ_p	2^{20}	$\lvert\epsilon_p\rvert < 4.77 \times 10^{-7}$
γ_{trig}	2^{20}	$\lvert\epsilon_{trig}\rvert < 4.77 \times 10^{-7}$
γ_π	2^{61}	$\lvert\epsilon_\pi\rvert < 2.17 \times 10^{-19}$
$\gamma_{\Delta t}$	2^5	$\lvert\epsilon_{\Delta t}\rvert < 0.03125$

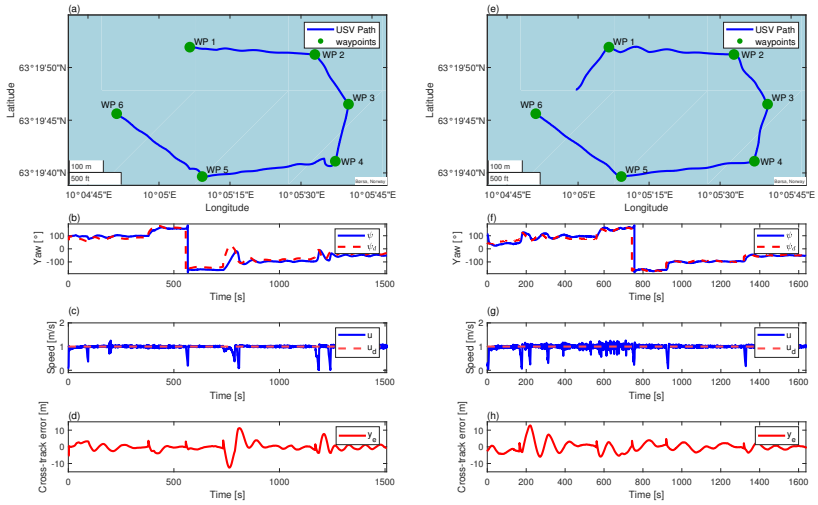


Figure 5.6 Experimental results from path 1. From experiment 1, (a)–(d) show the path of the USV plotted against the waypoints, the surge speed plotted against the desired surge speed, the yaw plotted against the desired yaw, and the cross-track error, respectively. From experiment 2, (e)–(h) show the path of the USV plotted against the waypoints, the surge speed plotted agains the desired surge speed, the yaw plotted against the desired yaw, and the cross-track error, respectively.

obtained using the first path, in Fig. 5.6, and we show the results from experiments 3 and 4, obtained using the second path, in Fig. 5.7. Links to videos of experiments 1 and 3 can be found in Appendix B.

5.8 Discussion

The experimental results successfully validated the proposed system and demonstrated that it is both computationally efficient and practical. In all four experiments, we initialized the vehicle somewhere between the origin in the NED frame and the

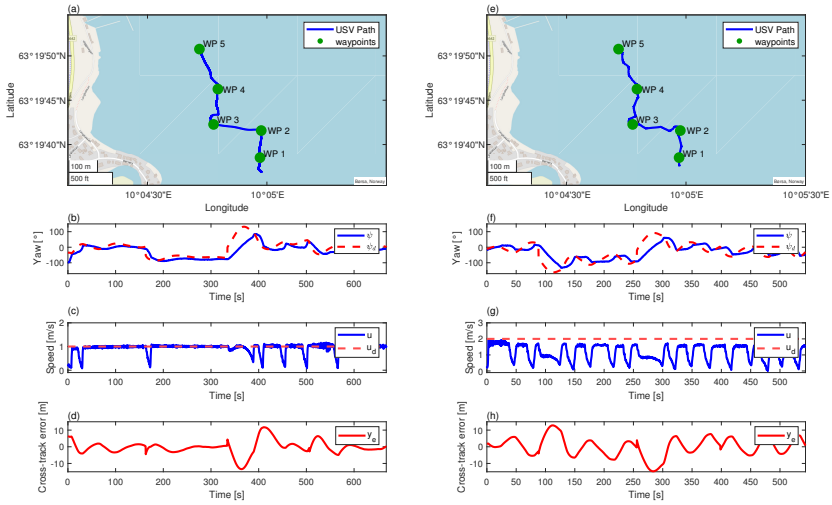


Figure 5.7 Experimental results from path 2. From experiment 3, (a)–(d) show the path of the USV plotted against the waypoints, the surge speed plotted against the desired surge speed, the yaw plotted against the desired yaw, and the cross-track error, respectively. From experiment 4, (e)–(h) show the path of the USV plotted against the waypoints, the surge speed plotted agains the desired surge speed, the yaw plotted against the desired yaw, and the cross-track error, respectively.

first waypoint. As a result, the initial position differs somewhat from experiment to experiment, most notably between experiments 1 and 2. We also see discontinuities in cross-track error when the vehicle reaches a waypoint. This is expected since the path-fixed frame also changes.

Except for two instances where the vehicle demonstrated inexplicable behavior, immediately following waypoints 4 and 5, we found that the parameters used in experiment 1 showed better performance than those used in experiment 2. Since the wind subsided between experiments 1 and 2, we would have expected to see better performance in experiment 2. In experiments 3 and 4, the path contains two sharp turns following waypoints 2 and 3. The parameters used in experiment 3 provided decent performance. However, when we increased the desired surge speed in experiment 4, we had to change the guidance parameters for the vehicle to follow the desired path. Nevertheless, the results obtained from experiment 4 are far from satisfactory, and the yaw controller disengaged the surge speed controller on several occasions in-between waypoints.

We found that the encrypted guidance system required very careful tuning of the parameters for good performance. Choosing a too-large integral gain, that is, k_i, significantly destabilizes the guidance system causing the vehicle to diverge off the path. To select an appropriate integral gain, the users should pay close attention to the dynamics of the system. In particular, vehicles with slow dynamics slowly

converge to the path, resulting in a significant integral windup. Moreover, we found that we had to relax k_i when the speed increases since an increase in speed results in a greater cross-track error, leading to integral windup. Whereas the analysis of (5.17) showed local stability properties, we do not get any information concerning the region of attraction by using Lyapunov's indirect method. However, we can infer an upper bound from the analysis of (5.12).

Concerning the problem of integral windup, we note that this problem is not unique to encrypted guidance systems. Indeed, all encrypted control systems with integral action will suffer from integral windup since the control systems have no means of checking their state. Similar restrictions are true for other nonlinear logic frequently used in feedback control systems. Another problem with the proposed guidance system is the initialization phase. Since the guidance algorithm is only locally stable, we must ensure that we initialize the vehicle within its stability region. Moreover, initializing with a significant cross-track error but within the region of stability results in an integral windup that will cause oscillations in cross-track error.

We note that we validated the system with integral action since the vehicle is equpped with a heading controller. Coupled with a course control system, we could have set $k_i = 0$, for which we have a stronger notion of stability, as discussed in Section 5.5.1.

Finally, the communication latency between the computer hosting the guidance system and the Cyberotter fluctuated between 80–300 ms during the experiments. In urban areas, we might expect the communication delay to decrease when 5G becomes fully operational. However, if used in rural areas, which might require different communication technologies, we might see an increase in communication latency. Nevertheless, guidance systems are usually characterized by relatively slow dynamics compared to the motion control system.

5.8.1 Drawbacks and possible improvements

The main drawback of the proposed method is that we only attain local stability. The global stability properties of conventional guidance algorithms arise from saturating functions, most notably the arctangent function, which is not readily available in ciphertext space. To make matters worse, the argument, a function of the cross-track error, can take infinitely large values. Therefore, there exists no computationally feasible approximation that can provide global stability. The same holds for other saturated functions, for example, the hyperbolic tangent function. As a result, we do not see any possibilities for developing encrypted guidance systems with global stability properties. Moreover, by using a linear approximation, to increase the stability region, we must reduce the control gains which results in less aggressive behavior and slower convergence for small cross-track errors, thus making the system more susceptible to disturbances. Using a more complicated guidance law, for example, the first three terms of a sine Taylor series approximation, we could increase the stability region while keeping a more aggressive convergence to the path for small cross-track errors. Such an alteration requires the support of homomorphic multiplications, which needs a fully homomorphic cryptosystem. As an interesting anecdote, we point out that while conventional guidance systems

typically seek to produce χ_d or ψ_d such that $\chi_d - \pi_p, \psi_d - \pi_p \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)$, encrypted guidance systems cannot implement such a mechanism. Hence, for large y_e^p, we can expect an encrypted guidance system to temporarily cause an increase in the along-track distance to its destination, or, in other words, 'backtrack' while reducing y_e^p. We observe this backtracking in Fig. 5.4, where we initialize the vehicle close to the border of stability in one of the simulations.

Another drawback of the proposed method is that we cannot keep the bearing π_p secret. We intentionally leak the bearing between each waypoint to enable rotations through multiplications with plaintext constants. If we wanted to keep π_p secret, we could use two approaches. The first method is a fully homomorphic cryptosystem that supports homomorphic multiplications with other ciphertexts. This approach would also allow keeping the guidance parameters k_p and k_i encrypted. Several fully homomorphic cryptosystems are available, but these are generally lattice-based cryptosystems whose security relies on the ring learning with errors problem by adding a 'small' noise to each plaintext. Homomorphic multiplications with other ciphertexts cause this noise to grow, after which we must use computationally expensive boot-strapping methods to reduce the noise. However, boot-strapping is probably not required with the limited number of homomorphic multiplications in an encrypted guidance system. As an added benefit, using either of these lattice-based cryptosystems makes the encrypted guidance system 'quantum secure'. The second option is to use labeled homomorphic encryption, which extends an additively homomorphic cryptosystem to allow homomorphic multiplications by revealing the mathematical operations by which a ciphertext is created. However, this approach seems impractical for encrypted guidance systems since we must send unique labels associated with each ciphertext, that is, each element of a rotation matrix and each waypoint coordinate, to the vehicle.

5.9 Chapter summary

We have presented a locally stable encrypted guidance system for straight-line path following that computes encrypted heading and course references using encrypted waypoints, encrypted position measurements, and the bearing between each waypoint. The motivation is to enable cloud-based guidance systems and guidance-as-a-service without revealing the position and path of the vehicle to the guidance provider. We implemented an encrypted guidance system on a USV and performed several field experiments. The results obtained from the field experiments show that the encrypted guidance system successfully guides the vehicle along a path consisting of straight-line segments between waypoints. We have discussed practical limitations that encrypted guidance systems face and proposed improvements that can increase the stability regions and improve the convergence rate of the proposed encrypted guidance systems.

Chapter 6

Revisiting Encrypted Fast Covariance Intersection

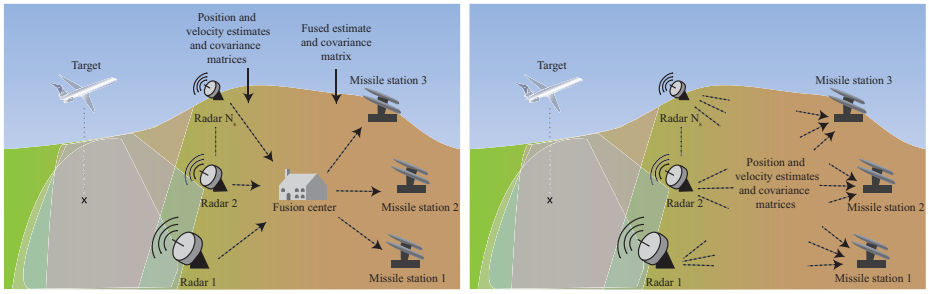
This chapter is based on the publication

[78] P. Solnør and A. G. Hem, "Revisiting encrypted fast covariance intersection," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–11, 2023. (Submitted)

6.1 Introduction

By promising unique benefits such as increased scalability, ease of maintenance, and software-as-a-service [32], cloud computing is attractive in numerous applications, such as sensor networks [176], smart grids [33], connected vehicles [177], and robotics [12, 34, 35]. However, by outsourcing the computation, we introduce new challenges concerning data privacy since the data passes through some third-party infrastructure. Passing unprotected data through a third-party infrastructure may lead to leaks of confidential information and constitutes an unacceptable breach of privacy for potential customers [178]. In particular, these concerns are relevant for situations where we want to fuse information produced by multiple heterogenous sources, for example, in multi-sensor information fusion.

Multi-sensor information fusion has numerous applications, including air traffic surveillance [179], autonomous driving [180], and general surveillance scenarios [181]. Early methods generally assumed that the state estimates produced by the different sensor systems were statistically independent. In practice, this assumption rarely holds [182]. Hence, new algorithms that consider the correlation between estimates followed. However, knowing or finding the exact correlation between estimates is often very difficult [183, Ch. 8]. A different problem that often arises is double-counting, that is, using an input estimate multiple times in the fusion. The result is that the fused output skews toward the double-counted input. Both these problems with existing fusion algorithms motivated the development of the covariance intersection algorithm [184]. The covariance intersection method is robust against filter divergence by avoiding the need to know the correlation between the



(a) Centralized fusion architecture. (b) Decentralized fusion architecture.

Figure 6.1 A collaborative air defense system is a possible use-case for encrypted fast covariance intersection. (a) The original method by [68] and the accelerated variant presented in this chapter are both *centralized*. (b) Our second variant is *decentralized* in the sense that each recipient fuses and decrypts the data directly.

inputs and the problem of double-counting. In practice, we approximate the optimal solution to the covariance intersection problem. This approximation is called fast covariance intersection [185].

In this chapter, we consider the problem of fusing state estimates and covariances from N_s sensor systems without revealing estimates produced by the individual sensor systems. Moreover, we do not want to disclose the relative performance of the sensor systems. A previously proposed method achieves this by performing fast covariance intersection on encrypted data [68]. However, it suffers from two drawbacks; first, the algorithm requires arithmetic on integers of considerable size. Hence, it is computationally expensive. Second, the method is *centralized* since we must perform the fusion operations in a fusion center and assume that no recipient is dishonest.

We contribute by presenting two alternative variants, each addressing one of the above problems. We first describe an *accelerated* method that uses computationally efficient arithmetic operations over much smaller integers than the original method. Moreover, this size reduction reduces the amount of data transmitted by approximately 99%. In terms of power consumption, data transmission is expensive. Accordingly, in some potential applications, such as wireless sensor networks, such a reduction is of significant importance to maximize the lifetime of the sensors [186]. We proceed by presenting a *decentralized* method that permits the existence of dishonest recipients without leaking information from the remaining honest sensor systems. We show the centralized and decentralized variants in Fig. 6.1, where we consider a target-tracking scenario.

6.1.1 Related work

Several cryptographic and information-theoretic primitives have been used to construct privacy-preserving information fusion schemes, most notably secure multi-party computation, differential privacy, and homomorphic encryption. Secure multi-party computation is a cryptographic concept in which several entities collaborate to

evaluate a function while keeping their inputs private [37]. They are often inherently distributed systems and have been used to design privacy-preserving state estimation and information fusion schemes [38]. However, secure multi-party computation schemes are usually interactive and require bidirectional communication between the participants, which limits their applicability to real-time systems [40].

Differential privacy is another privacy-preserving technique that seeks to reveal statistics about a population or dataset while maintaining the privacy of each person or record [41]. Hence, we can use differential privacy to perform information fusion without leaking the input estimates. For example, Ny and Pappas [43] first used differential privacy to achieve privacy-preserving filtering in the Kalman filter setting. More recently, privacy-preserving filtering has been extended to the distributed Kalman filter setting, for example, by Moradi et al. [46] and Yan et al. [187], and to fault-tolerant distributed online estimation by Wang et al. [188]. The privacy results from adding an appropriate level of noise to the input data. As such, privacy comes at the cost of reduced accuracy of the fused output. In some applications, for example, the target-tracking scenario shown in Fig. 6.1, any reduction in the accuracy of the fused output is undesirable.

Finally, homomorphic encryption [36] denotes a set of cryptosystems where the encryption algorithm preserves some algebraic structure. This algebraic structure enables computations directly on encrypted data, an alluring property, for example, in cloud-computing settings. For instance, we can use homomorphic encryption to train neural networks on encrypted datasets [48] or convert pre-learned neural networks to accommodate the evaluation of encrypted inputs by producing an encrypted output [189]. Other applications include encrypted feedback control systems [57], vehicular guidance systems [77], and information fusion schemes [64]. In theory, we can use fully homomorphic cryptosystems to construct encrypted systems functionally equivalent to the corresponding unencrypted schemes. However, in practice, such realizations are infeasible from a computational point of view. Hence, the design of encrypted systems frequently boils down to finding ways to simplify the scheme such that real-time performance can be maintained without causing significant performance degradation, for example, in terms of the accuracy of the fused output of some encrypted fusion algorithm.

The first encrypted information fusion scheme was introduced by [64]. They presented an encrypted version of the information filter, an algebraically equivalent formulation of the Kalman filter, applicable if the system matrix is the identity matrix and the control matrix is null. As a result of these assumptions, the prediction step only consists of matrix additions, which they compute using an additively homomorphic cryptosystem. However, these assumptions hold for very few systems. Later, Alexandru and Pappas [65], and Zhang et al. [66] proposed two variants of static-gain state estimators using labeled homomorphic encryption and a hybrid homomorphic encryption scheme, respectively. Importantly, they are limited to static-gain formulations since the scaling factors that map real numbers to elements in the plaintext space cause the internal state of stateful implementations to blow up.

An encrypted variant of fast covariance intersection was first proposed by Ristic et al. [67]. The algorithm consists of a composition of an order-revealing and an additively homomorphic cryptosystem. While it fuses state estimates and covariance

matrices, they achieve this at the cost of leaking the relative performance of each sensor system. Moreover, it computes the weights using a series of comparisons, which results in approximation errors and does not scale well with many sensor systems. Ristic and Noack [68] recently enhanced this scheme by designing an encrypted fast covariance intersection algorithm that does not reveal the weights. Additionally, it only requires an additively homomorphic cryptosystem. The proposed algorithm is elegant and solves the problem of leaking the fusion weights by postponing the required multiplications from the fusion center to the decryption node.

6.1.2 Main contributions

We present two alternative variants of the encrypted fast covariance intersection algorithm presented by Ristic and Noack in [68]. The first method uses high-performance stream ciphers instead of asymmetric cryptography to perform the homomorphic operations. Through numerical simulations, we show that this constitutes an acceleration of five to six orders of magnitude. Moreover, we reduce the amount of data transmitted by approximately 99% for a 128-bit level of security, significantly lowering the required bandwidth and power consumption. The second method is a decentralized variant where the recipients fuse and decrypt the encrypted input estimates in a single operation using a cryptographic tool called privacy-preserving aggregation. The decentralized variant is attractive since it permits the existence of dishonest recipients. Both implementations are readily available to interested readers online.

6.1.3 Outline

We first introduce the covariance intersection principle in Section 6.2 before describing some additional notation and cryptographic concepts in Section 6.3 that we use throughout this chapter. We proceed by presenting the encrypted fast covariance intersection method by Ristic and Noack in Section 6.4. In Section 6.5, we describe our accelerated variant, and in Section 6.6, we present our decentralized variant. We then introduce a simulation study in Section 6.7 before we show our results in Section 6.8. In Section 6.9, we discuss the obtained results and potential drawbacks of the proposed methods. Section 6.10 concludes the chapter.

6.2 Covariance intersection

Covariance intersection is a widely used method for fusing Gaussian estimates [190], for example, from different sensors. It is robust because it is invariant to double counting and ensures consistency in the fused output. A consistent estimate is one where the covariance corresponds to the actual error between the estimate and the true value [191, p. 232]. Maintaining consistency is crucial to avoid divergence in algorithms such as Kalman filters. Given the uncertainty in the correlation between estimates from different sources, the covariance intersection method is often used in practical applications. In this section, we will present the principles of the covariance intersection method and the relevant mathematical formulations.

Suppose we have N_s sensor systems producing the unbiased estimates $\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{N_s}$ and the corresponding covariance matrices $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_{N_s}$. We then obtain the fused estimate $(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0)$ by solving the convex combination

$$\boldsymbol{P}_0^{-1} = \sum_{n=1}^{N_s} \omega_n \boldsymbol{P}_n^{-1} \tag{6.1}$$

$$\boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0 = \sum_{n=1}^{N_s} \omega_n \boldsymbol{P}_n^{-1}\hat{\boldsymbol{x}}_n, \tag{6.2}$$

where the weights should satisfy

$$\sum_{n=1}^{N_s} \omega_n = 1, \tag{6.3}$$

and $0 \leq \omega_i \leq 1$ holds. We compute the optimal set of weights ω_i by solving the nonlinear optimization problem

$$\min_{\omega_i} \ \mathrm{tr}(\boldsymbol{P}_0) = \min_{\omega_i} \mathrm{tr}\Big(\Big(\sum_{n=1}^{N_s} \omega_n \boldsymbol{P}_n^{-1}\Big)^{-1}\Big), \tag{6.4}$$

where $\mathrm{tr}(\boldsymbol{P})$ denotes the trace of matrix $\boldsymbol{P}$.

6.2.1 Fast covariance intersection

While yielding the optimal solution, (6.4) is computationally expensive to solve and hence, often impractical. Moreover, all combinations of weights that satisfy (6.3) are valid solutions, albeit not necessarily optimal. This flexibility provides the opportunity of using more practical suboptimal solutions. The most well-known of these approximations is the fast covariance intersection method, presented by Niehsen in [185]. By using the trace as a measure of the uncertainty of the different estimates, Niehsen further constrained the covariance intersection problem by demanding that

$$\mathrm{tr}(\boldsymbol{P}_n)\omega_n - \mathrm{tr}(\boldsymbol{P}_{n+1})\omega_{n+1} = 0 \ \forall \ n \in \{1, \ldots, N_s - 1\} \tag{6.5}$$

holds. Using this, we can compute the weights non-iteratively according to

$$\omega_i = \frac{\frac{1}{\mathrm{tr}(\boldsymbol{P}_i)}}{\sum_{n=1}^{N_s} \frac{1}{\mathrm{tr}(\boldsymbol{P}_n)}}, \tag{6.6}$$

and thus avoid the need for solving (6.4).

6.3 Elements from cryptography

We let $\mathbb{R}$ and $\mathbb{Z}$ denote the real numbers and the integers, respectively. Given $a, b \in \mathbb{Z}$, we let $a \mid b$ denote 'a divides b' and $\gcd(a, b)$ denotes the greatest common

divisor of a and b. If $\gcd(a, b) = 1$, we say that a is co-prime to b. For $n \in \mathbb{Z}$, we say that a is congruent to b modulo n if $n \mid (a - b)$ which we write as $a \equiv b \pmod{n}$. For any non-zero $n \in \mathbb{Z}$ and $a \in \mathbb{Z}$, we let $a \bmod n$ represent the smallest element in the set $\mathbb{Z}_n \coloneqq \{0, 1, ..., n-1\}$ congruent to a modulo n, and we let $\mathbb{Z}_n^\times \coloneqq \mathbb{Z}_n \setminus \{a \in \mathbb{Z}_n \mid \gcd(a, n) \neq 1\}$ denote the multiplicative group of integers modulo n. Finally, we let $\mathbb{Z}_2^k$ denote the set of k-length tuples $x = (x_{k-1}, \ldots, x_0)$ whose elements x_i belong to $\mathbb{Z}_2$. We will use the term *plaintext* to refer to unencrypted data, and the term *ciphertext* to refer to encrypted data. We use an uppercase C to denote a ciphertext whose subscript denotes the corresponding plaintext. When we use the term *cryptosystem*, we refer to a full specification of a cryptographic system meant to provide confidentiality, including a complete specification of the key space, the plaintext space, the ciphertext space, the encryption algorithm, and the decryption algorithm. Finally, we distinguish between cryptosystems where it is easy to derive the decryption key from the encryption key and cryptosystems where deriving the decryption key from the encryption key is computationally infeasible. We refer to the former as *symmetric*, and the latter as *asymmetric*.

For two algebraic groups $G = (X, \star), H = (Y, *)$ and a mapping $\phi\colon X \to Y$, we define a *group homomorphism* as in [152, p. 533].

Definition 10 (Group homomorphism)**.** *A homomorphism from a group $G = (X, \star)$ to a group $H = (Y, *)$ is a mapping $\phi\colon X \to Y$ such that $\phi(a \star a') = \phi(a) * \phi(a')$, $\forall\, a, a' \in X$.*

Informally, a mapping between two groups is a group homomorphism if it *preserves the group operation*. For some cryptosystems, the encryption function is a group homomorphism from the plaintext space to the ciphertext space. We say that these cryptosystems are *partially homomorphic*. If the group operation in the plaintext space corresponds to addition, we say that the cryptosystem is *additively homomorphic*.

Finally, the adversaries we consider in this chapter are *honest-but-curious* [13]. That is, the dishonest parties only seek to extract information we deem confidential, and do not actively tamper with or manipulate the data involved.

6.3.1 Stream ciphers

A stream cipher is a *stateful* symmetric encryption scheme. They work by extending a short secret key into a very long pseudorandom sequence, which we call the *keystream*, using a keyed finite state machine. This keystream should be computationally indistinguishable from a random sequence of bits. Most stream ciphers encrypt data by mixing the keystream with the plaintext through bitwise addition in $\mathbb{Z}_2$, that is, the exclusive-or operator, to obtain the ciphertext. Bitwise addition is frequently used because it is an involution and exceptionally simple to implement in hardware. It is also very efficient in software because all hardware architectures tend to provide an exclusive-or instruction. However, we can also use other invertible operations to mix the keystream with the plaintext. An example of such a pair is modular addition and subtraction in $\mathbb{Z}_n$, where n is arbitrary. Proposition 1 shows that such additions are secure.

Proposition 1. *Let $X, Y \in \mathbb{Z}_n$ be two random variables where X is uniformly distributed and Y is independent of X. Then $X + Y \bmod n$ is uniformly distributed.*

Proof. We abuse notation slightly and let all additions inside the parentheses be performed modulo n. Then, for any $Z \in \mathbb{Z}_n$, we have

$$\begin{aligned}\Pr(X + Y = Z) &= \sum_{i=0}^{n-1} \Pr(X + Y = Z \mid X = i)\Pr(X = i) \\ &= \sum_{i=0}^{n-1} \Pr(Y + i = Z \mid X = i)\frac{1}{n} \\ &= \frac{1}{n}\sum_{i=0}^{n-1} \Pr(Y = Z + i) = \frac{1}{n}.\end{aligned}$$

□

6.3.2 Privacy-preserving aggregation

Privacy-preserving aggregation is a different cryptographic technique used to compute an aggregate sum from several encrypted summands [192]. It resembles secure multi-party computation in some aspects. However, it is generally *non-interactive* in that information only flows from the contributors providing the encrypted summands to the aggregators, that is, the intended recipients. The security notion of a privacy-preserving aggregation scheme is that of *aggregator obliviousness*, which, informally, requires that an aggregator only learns the aggregated sum and nothing of the contribution of each summand. Moreover, anyone not authorized to have the aggregated sum should be incapable of inferring meaningful information from the encrypted summands.

A privacy-preserving aggregation scheme consists of a tuple of three algorithms, (Setup, Enc, AggrDec), where the Setup-algorithm generates N_s private keys $\mathrm{sk}_1, \ldots, \mathrm{sk}_{\mathrm{N_s}}$, which are distributed to the contributors, and the aggregation key sk_0, which is distributed to the aggregators. Typically, the keys are chosen such that $\mathrm{sk}_0 = -\sum_{i=1}^{N_s} \mathrm{sk}_i$ [192, 193] or some equivalent variant [194]. The Enc-algorithm is used by the contributors to encrypt data using their private keys, while the AggrDec-algorithm is used to compute an aggregated plaintext sum from N_s ciphertexts using the aggregation key sk_0.

6.3.3 Obtaining valid plaintexts

The plaintext spaces of the cryptosystems we deal with in this chapter consist of some commutative ring of integers $\mathbb{Z}_n$. However, the state estimates, the covariance matrices, and the weights come from some real interval $\mathcal{I} = [a, b] \subset \mathbb{R}$. Hence, we must map these values to elements in the plaintext space. To this end, we use the mapping $\rho\colon \mathcal{I} \times \mathbb{Z}_n \to \mathbb{Z}_n$, defined as

$$\rho(x, \gamma) \coloneqq \begin{cases} \lceil \gamma x \rfloor, & \text{if } x \geq 0 \\ \lceil \gamma x \rfloor + n, & \text{otherwise,} \end{cases} \tag{6.7}$$

where γ is a scaling factor and $\lceil \cdot \rfloor$ denotes rounding to the nearest integer. Once we recover a plaintext following the homomorphic operations, we use the inverse mapping $\rho^{-1} \colon \mathbb{Z}_n \times \mathbb{Z}_n \to \mathcal{I}$, defined as

$$\rho(m, \gamma)^{-1} \coloneqq \begin{cases} \frac{m-n}{\gamma}, & \text{if } m \geq \frac{n}{2} \\ \frac{m}{\gamma}, & \text{otherwise,} \end{cases} \tag{6.8}$$

to recover the appropriate output without scaling. To simplify the presentation and notation, we assume that variables have been mapped to appropriate plaintexts using (6.7) before encryption and are mapped back to $\mathcal{I}$ using (6.8) once decrypted.

6.4 Encrypted fast covariance intersection

Ristic and Noack [68] proposed a variant of fast covariance intersection in encrypted form using homomorphic encryption. The privacy goal is to deny the fusion center any information about the individual input and output estimates, including the weight assigned to each state estimate. Their idea is to substitute (6.6) into (6.1)–(6.2), which yields

$$\boldsymbol{P}_0^{-1} = \Big(\sum_{i=1}^{N_s} \frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\Big)^{-1} \sum_{i=1}^{N_s} \frac{1}{\operatorname{tr}(\boldsymbol{P}_i)} \boldsymbol{P}_i^{-1} \tag{6.9}$$

$$\boldsymbol{P}_0^{-1} \hat{\boldsymbol{x}}_0 = \Big(\sum_{i=1}^{N_s} \frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\Big)^{-1} \sum_{i=1}^{N_s} \frac{1}{\operatorname{tr}(\boldsymbol{P}_i)} \boldsymbol{P}_i^{-1} \hat{\boldsymbol{x}}_i. \tag{6.10}$$

We can compute the sums $\sum_{i=1}^{N_s} 1/\operatorname{tr}(\boldsymbol{P}_\mathrm{i})$, $\sum_{i=1}^{N_s} (1/\operatorname{tr}(\boldsymbol{P}_i)) \boldsymbol{P}_i^{-1}$, and $\sum_{i=1}^{N_s} (1/\operatorname{tr}(\boldsymbol{P}_i)) \boldsymbol{P}_i^{-1} \hat{\boldsymbol{x}}_i$ using an additively homomorphic cryptosystem. To this end, Ristic and Noack [68] use the Paillier cryptosystem, whose ciphertext space is $\mathbb{Z}_{N^2}^\times$, where $N = pq$ is a factoring modulus consisting of two primes, p and q, of equal size in bits. The security of the Paillier cryptosystem rests on the assumption that for an integer x, determining if there exists an integer y such that $y = x^N \bmod N^2$ holds is hard. We refer to this assumption as the *decisional composite residuosity assumption*. For more details on the Paillier cryptosystem, we refer to [52]. Assuming we have an N_d-dimensional state vector, the recipients must decrypt the $N_d \times N_d$ fused matrix $\sum_{i=1}^{N_s} (1/\operatorname{tr}(\boldsymbol{P}_i)) \boldsymbol{P}_0^{-1}$, the fused vector $\sum_{i=1}^{N_s} (1/\operatorname{tr}(\boldsymbol{P}_i)) \boldsymbol{P}_0^{-1} \hat{\boldsymbol{x}}_0$ with N_d elements, and the fused scalar $\sum_{i=1}^{N_s} 1/\operatorname{tr}(\boldsymbol{P}_i)$. Finally, the recipients recover $\boldsymbol{P}_0^{-1}$ and $\boldsymbol{P}_0^{-1} \hat{\boldsymbol{x}}_0$ using element-wise multiplications with $(\sum_{i=1}^{N_s} 1/\operatorname{tr}(\boldsymbol{P}_i))^{-1}$.

For each timestep, the sensor systems and the recipients perform $N_d^2 + N_d + 1$ iterations of the Paillier encryption and decryption functions, respectively. The fusion center performs $N_s(N_d^2 + N_d + 1)$ homomorphic additions. From a computational perspective, the encryption and decryption operations consist of expensive exponentiations with large moduli. Moreover, the Paillier cryptosystem results in a massive ciphertext expansion since it maps k-bit plaintexts to $\log_2 N^2$-bit ciphertexts, where $k = 64$ often provides sufficient precision and $\log_2 N^2 = 6144$ corresponds to 128-bit security [160, Table 2]. Homomorphic additions in the Paillier cryptosystem consist of modular multiplications of the ciphertexts. Depending on

the computer architecture[1], the current state-of-the-art multiplication algorithms for numbers on the order of 4096-6144 bits are the Karatsuba [195] and the Toom-Cook [196, 197] multiplication algorithms with asymptotic runtimes of $O(n^{1.585})$ and $O(n^{1.465})$, respectively, where n denotes the size of the operands in bits. Hence, performing multiplications on elements in $\mathbb{Z}_{N^2}^{\times}$, that is, homomorphic additions, is computationally expensive. The second undesirable property of the proposed architecture in [68], as shown in Fig. 6.1a, is that the recipients have the decryption keys of the individual sensor systems. Hence, we must assume that no recipient of the fused estimates is dishonest and collaborates with the fusion center.

6.5 Accelerated encrypted fast covariance intersection

We start by presenting an accelerated variant of the scheme in [68], using the observation that Ristic and Noack assume that no recipient collaborates with the fusion center. Taking advantage of this assumption, we can switch to techniques from symmetric cryptography, which are considerably faster than the Paillier cryptosystem. Suppose we use a stream cipher to generate a pseudorandom keystream. We can then extract a k-bit tuple from the keystream, $(x_{k-1}, \dots, x_0)$, and interpret it as an unsigned integer using the bijective mapping $\psi\colon \mathbb{Z}_2^k \to \mathbb{Z}_{2^k}$, defined as $\psi(x_{k-1}, \dots, x_0) \coloneqq \sum_{i=0}^{k-1} x_i 2^i$.

Proposition 2. *Let $X = (x_{k-1}, \dots, x_0) \in \mathbb{Z}_2^k$ be generated by a stream cipher such that X is uniformly distributed. Then $Y = \psi(X) \in \mathbb{Z}_{2^k}$ is uniformly distributed.*

Proof. Since X is uniformly distributed and ψ is bijective, it follows that Y is uniformly distributed. □

From Proposition 1 and 2, it follows that we can use modular addition with a pseudorandom integer in $\mathbb{Z}_{2^k}$, generated using a stream cipher, to encrypt a plaintext value in $\mathbb{Z}_{2^k}$. We say that the uniformly distributed integer extracted from the keystream *additively blinds* the plaintext. Decryption follows by modular subtraction of the same pseudorandom integer. It is trivial to show that the proposed scheme is additively homomorphic. Consider the following example. Let $m_i \in \mathbb{Z}_{2^k}$ denote a plaintext and $s_i \in \mathbb{Z}_{2^k}$ a one-time key extracted from the keystream generated by user i, where $i \in \{1, \dots, N_s\}$. The users generate the keystreams using N_s instantiations of a stream cipher with different keys. We can then perform homomorphic additions as

$$\sum_{i=1}^{N_s} c_i = \sum_{i=1}^{N_s} m_i + s_i \bmod 2^k \tag{6.11}$$

$$= (\sum_{i=1}^{N_s} m_i + \sum_{i=1}^{N_s} s_i) \bmod 2^k, \tag{6.12}$$

[1] In practice, constant terms arising from, for example, the cache architecture, the multiplication instruction, and the interpolation procedure are significant. Hence, the fastest multiplication algorithm for integers of a given size is usually found empirically by comparing these algorithms.

Algorithm 16 Accelerated Encrypted Fast Covariance Intersection

Sensor system i

1: Compute $\frac{1}{\text{tr}(\boldsymbol{P}_i)}$, $\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}$, and $\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i$

2: Iterate a stream cipher initialized with secret key i to generate a keystream

3: Additively blind each element of $\frac{1}{\text{tr}(\boldsymbol{P}_i)}$, $\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}$, and $\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i$ to obtain $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}}$, $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}}$, and $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i}$

4: Send $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}}$, $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}}$, and $C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i}$ to the fusion center

Fusion center

5: Compute $C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}} \leftarrow \sum_{i=1}^{N_s} C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}}$, $C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}} \leftarrow \sum_{i=1}^{N_s} C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}}$, and $C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0} \leftarrow \sum_{i=1}^{N_s} C_{\frac{1}{\text{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i}$

6: Send $C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}}, C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}}$, and $C_{\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0}$ to the recipients

Recipient

7: Iterate N_s stream ciphers to generate the cumulative keystream

8: Recover $\frac{1}{\text{tr}(\boldsymbol{P}_0)}$, $\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}$, and $\frac{1}{\text{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0$ by element-wise subtraction of the cumulative keystream

9: Compute $\boldsymbol{P}_0^{-1}, \boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0$ by element-wise multiplication with $(\frac{1}{\text{tr}(\boldsymbol{P}_0)})^{-1}$

and it follows that we recover the sum $\sum_{i=1}^{N_s} m_i$ by computing

$$\sum_{i=1}^{N_s} m_i = (\sum_{i=1}^{N_s} c_i - \sum_{i=1}^{N_s} s_i) \bmod 2^k. \tag{6.13}$$

Correctness follows if $\sum_{i=1}^{N_s} m_i < 2^k$, which is trivial to ensure by choosing appropriate values for γ and k. Importantly, we now deal with arithmetic in $\mathbb{Z}_{2^k}$ instead of $\mathbb{Z}_{N^2}^{\times}$. It will often suffice to let k correspond to the word size of the processor, in which case we can represent all variables as unsigned integers held in individual general-purpose registers. Adding such integers takes a single cycle of the CPU, which is very fast. Moreover, assuming the number of bits used to represent the original floating-point data is equal to the number of bits used to represent the unsigned integers, there is no ciphertext expansion. Hence, the amount of data transmitted is significantly reduced compared to the scheme described in [68]. We summarize the steps in Algorithm 16, where we perform all additions modulo 2^k.

Remark. *Most stream ciphers are synchronous. Assuming we use a synchronous stream cipher to generate the keystreams, we must actively synchronize the keystreams between the sensor systems and the recipients. We achieve synchronization through conventional mechanisms such as IVs. To keep the presentation concise, we omit such implementation details.*

6.5.1 Security model and analysis

Similar to the original scheme by [68], the recipients hold the decryption keys. Hence, we must assume that no decryption nodes, for example, the missile stations

in Fig. 6.1a, are dishonest and share the decryption key with the fusion center. Moreover, since we use symmetric stream ciphers, we know that sensor system i holds s_i, which is part of the decryption key. Thus, we can consider situations where $1 \leq n \leq N_s - 1$ sensor systems, for example, the radar stations in Fig. 6.1a, are dishonest and collaborate with the fusion center to uncover the contribution from the remaining honest sensor systems. We let $\mathcal{S} \coloneqq \{1, \ldots, N_s\}$ denote the set of sensor systems participating in the fusion and $\mathcal{C} \subset \mathcal{S}$ the subset of dishonest sensor systems. If the dishonest sensor systems share their one-time keys, $s_i \; \forall \; i \in \mathcal{C}$, the fusion center is left with

$$c = \sum_{i=1}^{N_s} m_i + \sum_{i \in \mathcal{S} \setminus \mathcal{C}} s_i. \tag{6.14}$$

Since $\mathcal{S} \setminus \mathcal{C} \neq \emptyset$, it follows that $\sum_{i=1}^{N_s} m_i$ remains additively blinded by $|\mathcal{S} \setminus \mathcal{C}| \geq 1$ one-time keys. It follows from Proposition 1 that the dishonest sensor systems and the fusion center do not gain any information about the contribution of the honest sensor systems, $m_i \; \forall \; i \in \mathcal{S} \setminus \mathcal{C}$, and the fused output, $\sum_{i=1}^{N_s} m_i$, except for the subsum $\sum_{i \in \mathcal{C}} m_i$ produced by the dishonest sensor systems. Hence, the accelerated scheme's security follows the security of the underlying stream cipher.

6.6 Decentralized encrypted fast covariance intersection

With the approach in [68] and the accelerated method presented in Section 6.5, we must assume that none of the recipients of the fused estimates are dishonest and that the fusion center and the recipients are distinct entities. Otherwise, a dishonest recipient can share the decryption key with which the fusion center can decrypt the individual estimates and obtain the weights assigned to each sensor system. However, in practice, it is reasonable to imagine situations where a recipient and a sensor system belong to the same entity. From a 'structural' point of view, it would be nice to design a scheme where this assumption is not needed.

One way to avoid this assumption is by removing the fusion center altogether, as shown in Fig. 6.1b. Since (6.9)–(6.10) only consists of sums, we can use privacy-preserving aggregation to construct a decentralized variant where we eliminate the need for a fusion center. The idea is to use the privacy-preserving aggregation scheme to perform element-wise aggregation and decryption. In the context of Section 6.3.2, the sensor systems are the contributors, while the recipients are the aggregators. Each recipient holds the aggregation key sk_0 and receives N_s encrypted traces, state estimates, and covariance matrices from the sensor systems. It performs element-wise fusion and decryption using AggrDec. Algorithm 17 outlines the steps.

6.6.1 Security model and analysis

In the decentralized setup, we can imagine a situation where dishonest recipients and sensor systems collaborate. Clearly, if we allow $N_s - 1$ dishonest sensor systems and at least one dishonest recipient, they can recover the private key of the remaining

Algorithm 17 Decentralized Encrypted Fast Covariance Intersection

Sensor system i

1: Compute $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}$, $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}$, and $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i$

2: Perform element-wise encryption of $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}$, $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}$, and $\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i$ to obtain $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}}$, $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}}$, and $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i}$

3: Send $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}}$, $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}}$, and $C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_i)}\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i}$ to the recipients

Recipient

4: Compute $\frac{1}{\operatorname{tr}(\boldsymbol{P}_0)} \leftarrow \textsc{AggrDec}(\mathrm{sk}_0, C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_1)}}, \ldots, C_{\frac{1}{\operatorname{tr}(\boldsymbol{P}_{N_s})}})$

5: Recover $\frac{1}{\operatorname{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}$ and $\frac{1}{\operatorname{tr}(\boldsymbol{P}_0)}\boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0$ by element-wise aggregation and decryption using AggrDec

6: Compute $\boldsymbol{P}_0^{-1}, \boldsymbol{P}_0^{-1}\hat{\boldsymbol{x}}_0$ by element-wise multiplication with $(\frac{1}{\operatorname{tr}(\boldsymbol{P}_0)})^{-1}$

honest sensor system by computing $s_{j\in\mathcal{S}\setminus\mathcal{C}} = s_0 + \sum_{i\in\mathcal{C}} s_i$. Moreover, the dishonest actors can subtract the contributions of the dishonest sensor systems from the fused output to identify the contribution of the remaining honest sensor system. However, if we consider $1 \leq n \leq N_s - 2$ dishonest sensor systems, they can only uncover the combined contribution of the remaining $|\mathcal{S} \setminus \mathcal{C}| \geq 2$ honest sensor systems. Hence, the security of the decentralized scheme follows from the aggregator obliviousness property of the underlying privacy-preserving aggregation scheme.

6.7 Simulation study

We test the accelerated and decentralized variants against the method by Ristic and Noack [68] in a target-tracking scenario. The simulation study's purpose is to assess the performance of the different instantiations in terms of the computational latency induced for a given level of security. Moreover, we want to validate that applying the proposed encryption methods does not adversely affect the accuracy of the fused output.

We use the Rabbit stream cipher [105] to generate the keystreams for the accelerated variant. The Rabbit stream cipher is attractive since it is both computationally efficient in software [74] and has a relatively small internal state. We prefer a small internal state because the recipients must iterate N_s instantiations of the stream cipher to generate the decryption keys. Large internal states may not fit in low-level caches, and accessing higher-level memory can introduce considerable access latencies. If N_s is large, the resulting overhead could be significant. Since the Rabbit stream cipher provides 128-bit security, we use a 3072-bit factoring modulus in the Pailler instantiation of the Ristic and Noack scheme corresponding to approximately 128-bit security [160, Table 2].

For the decentralized variant, we use the privacy-preserving aggregation scheme by [193], which is also based on the decisional composite residuosity assumption. We measure the induced computational latency for 1024-, 2048-, and 3072-bit factoring moduli, corresponding to approximately 64-, 112-, and 128-bit security margins, respectively [160, Table 2]. We implement all algorithms in C++, using the Rabbit

Table 6.1 System specifications used for the simulations

Hardware	
Model name	Dell Precision 5550
CPU	Intel Core i7 10850H
Instruction set architecture	x86-64
Number of cores	6
Frequency	2.70 GHz
Word size	64-bit
Memory	32 GB
Software	
Operating system	Ubuntu 20.04 LTS
Big number library	GMP 6.2.1
Compiler	g++ 9.4.0

implementation from the CryptoToolbox [79] and the GNU Multiple Precision Arithmetic Library [162] for big-number arithmetic and number-theoretic functions.

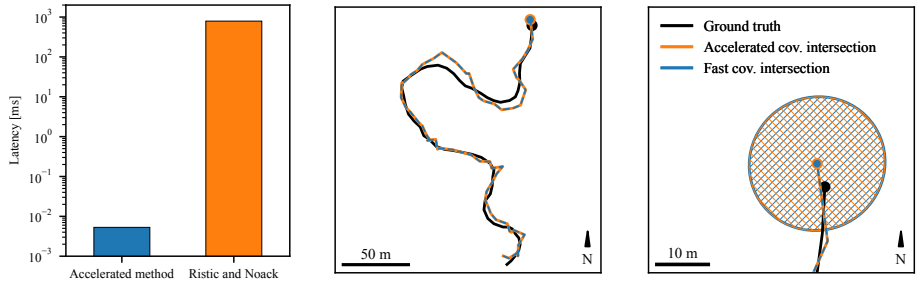
6.7.1 Datasets

We test the algorithms on datasets consisting of a target whose movement we model using discrete white noise acceleration (DWNA) [191, p. 273] with the process noise set to 0.6 [m/s^2]. The target is detected by N_s radar stations evenly spaced around a circular surveillance area with a radius of 1000 meters. The radar stations propagate the Cartesian position and velocity estimates using the DWNA model and update them using a Kalman filter based on the measurements. The datasets include the updated estimates and their covariances.

The model used to generate the measurements in the datasets includes both polar and Cartesian noise, representing errors in bearing and range and other errors like clustering inaccuracies and receiver noise, respectively. The measurement noise matrix, $\boldsymbol{R}$, is defined as $\boldsymbol{R} = \boldsymbol{R}_c + \boldsymbol{J}\boldsymbol{R}_p\boldsymbol{J}^\top$, where $\boldsymbol{J}$ is the Jacobian of the mapping from polar to Cartesian coordinates, and $\boldsymbol{R}_c$ and $\boldsymbol{R}_p$ are defined as

$$\boldsymbol{R}_c = \begin{bmatrix} \sigma_{\text{xy}}^2 & 0 \\ 0 & \sigma_{\text{xy}}^2 \end{bmatrix} \text{ and } \boldsymbol{R}_p = \begin{bmatrix} \sigma_{\text{r}}^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}. \tag{6.15}$$

The values for σ_{xy}, σ_{r}, and σ_θ are set to 10 [m], 8 [m], and 1.0 [deg], respectively. We run all the simulations on a portable workstation, the specifications of which we list in Table 6.1. We set the scaling factor γ to 2^{40} and define the plaintext spaces as $\mathbb{Z}_{2^{64}}$. For the Rabbit stream cipher, we also define the ciphertext space as $\mathbb{Z}_{2^{64}}$, meaning that we can use 64-bit unsigned integers to represent all plaintexts and ciphertexts.



(a) (b) (c)

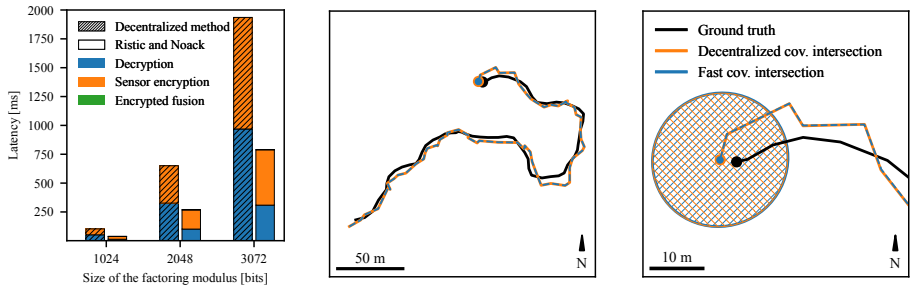
Figure 6.2 Results from the accelerated algorithm, where (a) shows the computational latencies induced by our accelerated variant instantiated with the Rabbit stream cipher [105] against the method by [68] instantiated with a 3072-bit factoring modulus, (b) shows the fused track from our accelerated variant, conventional fast covariance intersection, and the ground-truth, and (c) shows the covariance ellipses from our accelerated variant and conventional fast covariance intersection.

6.8 Results

We present the results by comparing the computational latencies induced by the accelerated and decentralized variants relative to the scheme described by [68]. Moreover, we show the fused tracks of the target produced by the accelerated and decentralized variants compared to the ground truth and the output produced by the conventional, unencrypted fast covariance intersection method. Finally, we show the fused covariance ellipses from the accelerated and decentralized variants and compare them with the covariance ellipses from the conventional, unencrypted fast covariance intersection method. In the simulations, we fuse the output from $N_s = 10$ radar stations. We present the results of the accelerated variant in Fig. 6.2 and the decentralized variant in Fig. 6.3. The implementations of the algorithms and additional datasets containing 2, 4, 6, 8, 10, and 15 radar stations are available online [198].

6.9 Discussion

Figure 6.2 shows that the fusion result of the accelerated variant is identical to the output of a conventional, unencrypted fast covariance intersection algorithm in terms of the accuracy of the fused estimates. This is intuitive since the quantization error is negligible. Moreover, while not shown in the figure to avoid unnecessary clutter, we note that the fused output is identical to the result of the original method by [68]. This is also expected since we use similar precision to represent the plaintexts. Figure 6.2a shows the computational latencies induced by the accelerated variant and the original method. The results show that the accelerated variant is five to six orders of magnitude faster than the original method. Moreover, our target-tracking datasets consist of four-dimensional state vectors. Since the number of homomorphic



(a) (b) (c)

Figure 6.3 Results from the decentralized algorithm, where (a) shows the computational latencies induced by our decentralized variant instantiated with the privacy-preserving aggregation scheme by [193] against the method by [68], (b) shows the fused track from our decentralized variant, conventional fast covariance intersection, and the ground-truth, and (c) shows the covariance ellipses from our decentralized variant and conventional fast covariance intersection.

additions is quadratic in the dimension of the state vector, we expect the benefit of using the accelerated variant to increase with higher-dimensional state vectors. By using the accelerated variant, we also reduce the size of each ciphertext, and hence the total amount of data transmitted, by $1 - 64/6144 \approx 99\%$ compared to the original scheme. A drawback of the accelerated method is that the decryption node must iterate N_s stream ciphers to produce $\sum_{i=1}^{N_s} s_i \bmod 2^k$ to recover the sum of the plaintexts. Therefore, the proposed acceleration does not scale well with many sensor systems, that is, large N_s. Contrary to the original scheme, the recipient must be informed of sensor systems not participating in the fusion to recover the fused estimate. Moreover, if we add a new sensor system, we must distribute the symmetric key of the newly added sensor system to all recipients.

Figure 6.3 shows that the fusion result of the decentralized variant is also identical to the output of conventional, unencrypted fast covariance intersection in terms of the accuracy of the fused estimates. This is expected because we operate in the same plaintext space as we did for the accelerated variant. Figure 6.3a shows the computational latencies induced by the decentralized variant and the original method by [68] using three configurations with different factoring moduli. The decentralized variant using privacy-preserving aggregation appears to be slow compared to the original scheme using the Paillier cryptosystem, despite both being based on the decisional composite residuosity assumption and consisting of seemingly similar operations. There are two reasons for this discrepancy. First, we can use the Chinese remainder theorem to accelerate the Paillier decryption algorithm by computing the exponentiations modulo p^2 and q^2 instead of modulo N^2 which reduces the computational latency induced by a factor of approximately $1/2$. We cannot achieve a similar acceleration for the privacy-preserving aggregation scheme because giving the aggregator access to the factorization of N invalidates the decisional composite residuosity assumption. Hence, we would lose the aggre-

gator obliviousness property, and the recipients, in possession of the ciphertexts corresponding to the individual estimates, could derive information concerning these estimates. Second, the exponents used in the encryption and aggregated decryption algorithms of the privacy-preserving aggregation scheme, sk_i, are elements with $2 \log_2 N^2$ bits, whereas the exponents used in the Paillier encryption and decryption algorithms, N and $\lambda = (p-1)(q-1)$, respectively, are elements with $\log_2 N$ bits. This constitutes an additional acceleration of the Paillier algorithms compared to the privacy-preserving aggregation algorithms. If we want to add or remove a sensor system, we must generate a new set of keys sk_i by running the SETUP-algorithm and then distribute the updated keys to the sensor systems and the aggregators. For the original scheme, removing a sensor has no implications for the correctness of the computation, while adding a new sensor system consists of sending the public key to the newly added sensor system.

6.10 Chapter summary

We have presented two variants of encrypted fast covariance intersection. The first scheme is an accelerated variant that uses computationally efficient stream ciphers instead of asymmetric cryptography. We show that the accelerated variant is five to six orders of magnitude faster than the original scheme through simulations. Moreover, the accelerated variant decreases the amount of data transmitted by approximately 99% compared to the original scheme when instantiated with 128-bit security, significantly reducing the required bandwidth and power consumption. The acceleration does not require different security assumptions, nor does it adversely affect the accuracy of the fused output. The second scheme is a decentralized variant where the recipients use privacy-preserving aggregation to fuse and decrypt the estimates directly. We argue that the decentralized scheme is attractive because it permits the existence of dishonest recipients without disclosing the state estimates and the relative performance of the individual sensor systems. We show that the computational latency induced by the decentralized variant is on the same order of magnitude as the original encrypted fast covariance intersection scheme. Moreover, the accuracy of the fused output is identical to the original scheme and conventional fast covariance intersection. The most significant drawback is that adding or removing sensor systems is more complicated than in the original scheme.

Chapter 7

Concluding Remarks

Throughout the thesis, several aspects of cybersecurity and applications of cryptographic methods in feedback control have been considered. Chapter 1 first introduced the subject and motivated the directions pursued by the individual chapters throughout the thesis. Part I of the thesis then focused on the vulnerability of transmission channels in NCSs in autonomous vehicles and efficient means of securing them without inducing intolerable computational latencies. In particular, the chapters considered using computationally efficient stream ciphers and authenticated encryption to make these vehicles more resilient against cyber-physical attacks. Part II of the thesis considered using homomorphic encryption to achieve privacy-preserving vehicular control and guidance systems before presenting two encrypted information fusion schemes. Emphasis has been placed on computational performance and experimental validation of the proposed methods. The purpose of this final chapter is to briefly summarize the contents of the thesis and its contributions before outlining some directions for future work.

7.1 Summary and discussion

We opened Part I of this thesis by considering the use of modern stream ciphers to enable fast and secure transmission of images and point clouds in NCSs in Chapter 2. The motivation for the research was the observation that previous publications proposed using various ad-hoc constructions to achieve authenticated encryption and mainly resorted to conventional block ciphers such as DES, 3DES, AES, and Blowfish. We argued that once initialized, the keystream generators of modern stream ciphers are more efficient than the full iteration of a block cipher without access to hardware-accelerated instructions and parallelization. Since images and point clouds are of considerable size, this could offset the overhead produced by the initialization phase of the stream ciphers. Therefore, we reasoned that stream ciphers are more appropriate for components in NCSs, where the available instruction sets may not provide hardware acceleration features of cryptographic operations. Moreover, we argued that cryptographically sound generic compositions or dedicated algorithms should be used to achieve authenticated encryption instead of the previously suggested ad-hoc constructions. We also considered the effect of using lossy and

lossless compression algorithms to reduce computational latency. To this end, we implemented a cryptographic pipeline in the ROS middleware and verified, through experiments on a dataset obtained from a USV, that modern software-oriented stream ciphers from the eSTREAM portfolio significantly outperform the AES block cipher, the de-facto standard encryption algorithm, when no hardware-accelerated instructions are available. Moreover, when hardware-accelerated instructions are available, the AEGIS stream cipher, a dedicated authenticated encryption algorithm that takes advantage of the AES round function, and can therefore take advantage of AES-specific instructions, significantly outperforms the AES. We also showed that the compression algorithms induce more significant computational delays than the cryptographic algorithms and should only be used if bandwidth is constrained.

In Chapter 3, we considered the practical implementation of several cyber-physical attacks against a USV and the implications of such attacks on the behavior of the USV. The motivation for the work was that most studies concerning cybersecurity in autonomous vehicles and NCSs are of a theoretical nature, where attacks against vulnerabilities are not necessarily described in detail. We discussed the use of IDSs versus using cryptographic methods to prevent such attacks from being successful and highlighted the problem of the base-rate fallacy when it comes to using anomaly-based IDSs in practical applications. In particular, this consideration is largely ignored by publications on the topic. Therefore, we argued that, when possible, cryptographic methods should be preferred. We demonstrated that we could use an attack vector known as ARP-spoofing to redirect navigation data to a malicious device that manipulated the position and heading measurements going to the guidance & control systems. We also demonstrated that authenticated encryption effectively prevent these attacks. Moreover, we showed that using auxiliary information such as timestamps is essential to detecting and preventing replay attacks. We demonstrated the effectiveness of the attacks and the defensive measures through several field experiments performed using the Cyberotter USV.

Moving into the second part of the thesis, we considered applications of homomorphic encryption. This work was mainly motivated by the observation that modern ICT allows outsourcing computations in NCSs to third-party cloud providers. Outsourcing computations can be desirable for several reasons; first, accessing software through subscription options can be more cost-efficient for companies than purchasing licensing rights outright. Moreover, outsourcing computations may reduce required on-site computational capacities, reducing costs related to infrastructure. Finally, hosting software on a centralized server eases installation and maintenance.

We started Part II of the thesis by designing a fully encrypted surge speed and heading control system for a USV in Chapter 4. The motivation for this chapter was that very few studies on encrypted control consider practical applications. Moreover, previous implementations and demonstrations were performed in controlled laboratory settings. Therefore, we wanted to implement an encrypted control system and demonstrate it on an experimental platform in an uncontrolled environment. To this end, we used a cryptosystem called labeled homomorphic encryption, which 'extends' additively homomorphic cryptosystems by enabling a certain number of homomorphic multiplications, to design a fully encrypted control system for the surge speed and yaw of a USV. Concerning which additively homomorphic cryptosystem to use, we suggested using the Joye-Libert cryptosystem instead the

more frequently used Paillier cryptosystem. Using the Joye-Libert cryptosystem should be beneficial since it results in half the ciphertext expansion of the Paillier cryptosystem. Since homomorphic additions of both these cryptosystems consist of multiplications in the ciphertext space, reducing the size of each ciphertext should significantly reduce the computational latency induced by these operations. The encrypted control system was implemented and ported to the Cyberotter USV, after which we validated the control system through field experiments in an uncontrolled environment. Hence, we consider the main contribution of this chapter to be the implementation and experimental validation of the encrypted control system in an uncontrolled environment, whereas previous studies have mainly resorted to simulation studies and experimental validations in controlled laboratory settings.

In Chapter 5, we proceeded by considering encrypted guidance systems. We were motivated by the observation that previous studies on encrypted control systems for vehicles only considered the control aspect, such as computing the thrust allocation to some actuator. However, outsourcing low-level control seems somewhat impractical if other components, such as the path-planning and guidance systems, run onboard the vehicle. Moreover, the inner-loop control systems tend to operate at much higher frequencies, and the transmission delays between the actuators and the controllers may cause significant deterioration of control performance. We then explained that the design of encrypted guidance systems poses different challenges than the design of encrypted control systems since we must deal with coordinate frames and trigonometric functions that are not directly available as homomorphic operations. Moreover, using approximations, for example, a Taylor series representation, may not be computationally feasible while maintaining real-time performance. To solve this, we first argued that we need not encrypt the general bearing between waypoints since inferring the position of a vehicle from this information alone is not feasible. Second, we argued that local stability properties might be sufficient in most applications since vehicles' tend to be 'close' to their desired paths. Hence, we could linearize the LOS and ILOS guidance laws, after which we could implement a semi-encrypted guidance system using an additively homomorphic cryptosystem. We showed that the linearized guidance laws possess desirable stability properties before implementing them in encrypted form. Through simulations, we also validated the hypothesis from Chapter 4 that, when requiring an additively homomorphic cryptosystem, using the Joye-Libert cryptosystem instead of the Paillier cryptosystem results in considerably faster implementations. Again, the effectiveness of the encrypted guidance system was tested and validated in closed loop in an uncontrolled environment using the Cyberotter USV. The main contributions of this chapter are the conceptualization and design of the encrypted guidance systems, the implementation, and the experimental validation.

Moving on to Chapter 6, we considered the problem of fusing unbiased Gaussian estimates produced by a heterogeneous set of sensors without revealing the individual estimates, the resulting fused output, and the weights assigned to individual estimates. The motivation for the work was a recently proposed scheme that achieves this using an additively homomorphic public-key cryptosystem. Following an initial description of the encrypted fast covariance intersection scheme, we showed that we could use symmetric-key stream ciphers instead of the public-key cryptosystem without changing the underlying security assumption. The motivation behind this

change was that symmetric-key stream ciphers are considerably faster than public-key cryptosystems. Moreover, by switching to symmetric-key stream ciphers, we avoided the ciphertext expansion associated with the public-key cryptosystem. After implementing the original scheme and the accelerated variant, we showed, through simulations on simulated datasets, that, for a 128-bit level of security, the change constitutes an acceleration of five to six orders of magnitude. The acceleration also reduced the amount of data transmitted by approximately 99% for a 128-bit level of security. In addition to the accelerated variant, we showed that we could use a cryptographic concept called privacy-preserving aggregation to implement a decentralized variant of the original scheme. The decentralized variant is attractive because we no longer need to assume that all recipients of the fused estimates are trusted.

A moving theme throughout the thesis has been the experimental validation of the proposed methods. The field experiments were largely possible thanks to the time invested in setting up the Cyberotter USV. While experimental validation requires considerably more time and effort than simulations in the office, they also provide additional insight into practical considerations of the proposed algorithms. For example, while the encrypted guidance laws in Chapter 5 showed local stability properties, the experimental validation showed that tuning the guidance gains appropriately such that the implementation was robust against environmental disturbances without becoming unstable was a considerable challenge. Such practical difficulties are easy to miss when resorting to simulations. The field experiments are, therefore, a significant part of what distinguishes the work presented in this thesis from previous work.

7.2 Future work

While there are several directions that future work can pursue, I want to highlight a couple of possibilities that I believe are particularly interesting. First, concerning the encrypted guidance system in Chapter 5, we used linearization of the LOS and ILOS guidance laws to enable implementations with the available homomorphic operations. In the following discussion, we described several means of increasing the region of convergence while maintaining relatively aggressive behavior for small cross-track errors. We claimed that one way of achieving this could be by using the first two terms of a Taylor approximation of a sinusoidal function instead of the linear approximation we used. However, an approach not considered in Chapter 5 is to defer the saturating operation, the arctangent function, to the vehicle. The inspiration for this idea is the observation that a similar deferral of a multiplication operation is used to achieve encrypted fast covariance intersection using an additively homomorphic cryptosystem. An interesting question is whether or not such a change can lead to global stability when we add integral action for heading control. We would typically use a guidance law that decreases the rate of integration for large cross-track errors to avoid the problem of integral windup. However, implementing such adaptive integral action in encrypted form seems challenging. It would also be interesting to consider encrypted guidance and control, where we host both systems remotely.

Concerning encrypted information fusion, it could be interesting to consider using threshold schemes to make the decentralized variant more fault-tolerant. However, using threshold schemes could result in reduced accuracy of the fused output since the recipient is unable to determine which encrypted state estimates to include and which encrypted state estimates to leave out when more than the required number of encrypted estimates are available. Moreover, the fusion of subsets of encrypted estimates may be used to derive information concerning the relative accuracy of individual estimates. Finally, we highlight that information fusion is only part of the larger picture. In practice, we must deal with data association to determine which estimates to fuse, an aspect we ignored in Chapter 6. Hence, a natural extension of the work would be to investigate whether we can design an encrypted data association algorithm.

Appendices

Appendix A

Experimental Platform

Throughout the work described in this thesis, I have contributed to developing the Cyberotter experimental platform in close collaboration with Øystein Volden and Kristoffer Gryte, researcher and head of the UAVlab at the Department of Engineering Cybernetics. The work builds on previous work, primarily from João Fortuna, a former Ph.D.-student, and Nikolai Lauvås, a current Ph.D.-student, both at the Department of Engineering Cybernetics. In addition, most of the work on the software stack builds on contributions from former students of the UAVlab, and the Underwater Systems and Technology Laboratory at the University of Porto, Portugal.

The purpose of developing the Cyberotter was to prepare a platform to facilitate the implementation and validation of algorithms through field experiments. The platform consists of two primary components; the USV and the land station. The land station serves two purposes. First, it includes a remote control station from

Figure A.1 Outdoor testing of the Cyberotter onboard navigation system in February 2021.

Figure A.2 Field experiment with the Cyberotter in the Trondheim Fjord in May 2022.

which we send commands and execute missions through the Neptus graphical user interface. Second, it consists of an RTK base station to enhance the accuracy of the position and velocity measurements. The RTK base station requires a survey-in period of approximately 18 hours, during which we needed access to electrical power. Hence, we primarily used the RTK base station to log datasets and for the experiments conducted at the Brattøra harbor, where we had access to power overnight from the offices of Maritime Robotics. Concerning the experimental validation of the encrypted control and guidance systems described in Chapters 4 and 5, respectively, these were conducted in Børsa. Because we did not have access to electrical power overnight and since we performed these experiments farther from shore, we did not use the RTK base station here.

Most of the developmental work on the Cyberotter was done during the winter and spring months of 2021, with some minor upgrades during the winter months of 2022. Figure A.1 shows the Cyberotter hardware during the initial testing of the onboard navigation system in February 2021. Figure A.2 shows the finished Cyberotter during one of the subsequent field experiments on the Trondheim Fjord in May 2022.

A.1 Hardware description

We show the current hardware schematic of the Cyberotter and the land station in Fig. A.3. The onboard navigation solution consists of two GNSS antennas, mounted at the stern and the bow, respectively, an ADIS IMU, two U-blox F9P GNSS receivers, and an SBG Ellipse 2D INS. We synchronize the IMU and the GNSS receivers using a Sentiboard [131]. Currently, we do not fuse the GNSS and the IMU measurements, but datasets with ground truth from the INS have been recorded. The Sentiboard passes data directly to the BeagleBone Black computer running the primary GNC system in DUNE. The INS sends data to a ROS node on an Nvidia Jetson Xavier. A software program bridges these ROS messages to IMC messages, which we send to the BeagleBone Black closing the loop. We show the specifications of the Nvidia Jetson Xavier, the BeagleBone Black, the IMU, the INS, and the GNSS receivers in Tables A.1, A.2, A.4, A.5, and A.3, respectively.

Concerning instrumentation, the Cyberotter is equipped with two stereo-camera configurations and a LiDAR. In addition, a rack and a connector for a doppler velocity log (DVL) have been mounted, although no DVL sensor is currently installed. The stereo-cameras, LiDAR, and the DVL pass data to an Nvidia Jetson Xavier embedded computer for signal processing in ROS.

Onboard communication consists of serial and Ethernet communication. The land station and the USV primarily communicate over a WiMAX link. We also set up two industrial 4G routers with dynamic domain name server to set up an IPsec tunnel between the USV and the office at Skippergata 14, Trondheim. We communicated over the IPsec tunnel for the closed-loop experiments with the encrypted control and guidance systems described in Chapters 4 and 5, respectively.

A.2 Software description

The software stack primarily consists of the LSTS toolchain consisting of DUNE, the IMC protocol, and Neptus, a graphical command and control interface. In addition, some features were implemented in ROS to take advantage of publicly available software packages and drivers. This includes the drivers for the INS, cameras, LiDAR, and the DVL. We configured the *imc_ros_bridge* [199], a software tool developed at Kungliga Tekniska Högskolan, Stockholm, to bridge IMC and ROS messages. We had to make some modifications to enable the conversion of custom ROS messages with encrypted data fields, message authentication tags, and IVs.

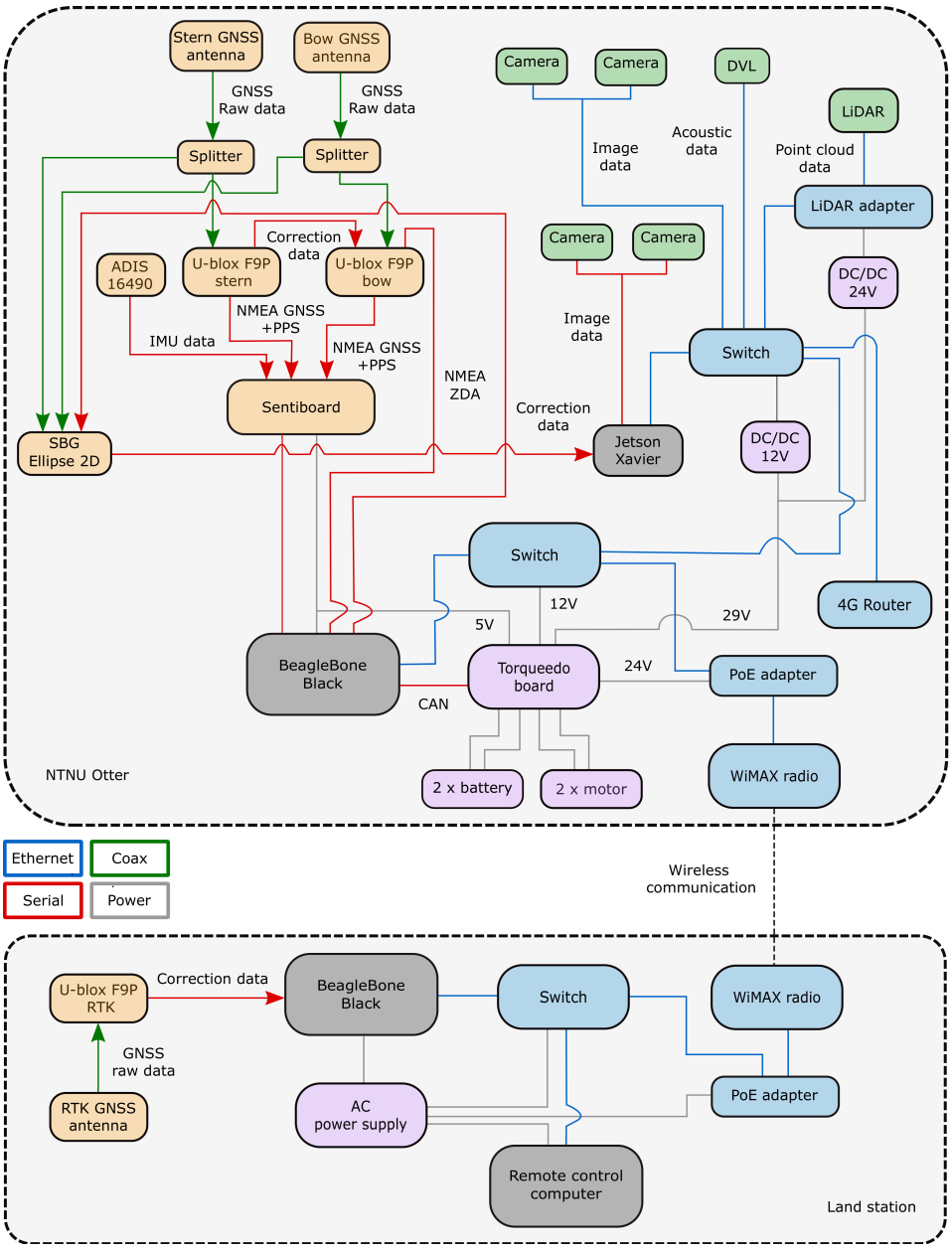


Figure A.3 Schematic of the Cyberotter and the associated land station.

Table A.1 Nvidia Jetson Xavier specifications

Model name	NVIDIA Jetson Xavier
CPU	NVIDIA Tegra Xavier
Instruction set architecture	ARMv8.2
Number of cores	8
Word size	64 bit
Memory	32GB
Operating system	Ubuntu 18.04 LTS

Table A.2 BeagleBone Black specifications

Model name	BeagleBone Black
CPU	Sitara AM3358BZCZ100
Instruction set architecture	ARMv7-A
Number of cores	1
Word size	32 bit
Memory	512 MB
Operating system	Custom Linux (Yocto)

Table A.3 GNSS receivers

Model name	U-blox F9P
GNSS constellations	GPS, GLONASS, Galileo, BeiDou
Interfaces	UART, SPI, USB, DDC
Update frequency	$7 - 25$ Hz
Operating temperature	-40°C to 85°C

Table A.4 Inertial measurement unit

Model name	Analog Devices ADIS 16490
Gyroscope	Triaxial
Gyro in-run bias stability	1.8°/h
Angular random walk	$0.09°/\sqrt{\text{h}}$
Accelerometer	Triaxial
Accel. in-run bias stability	3.6μg
Velocity random walk	$0.008\text{m/s}/\sqrt{\text{h}}$
Operating temperature	-40°C to 105°C

Table A.5 Inertial navigation system

Model name	SBG Ellipse 2-D
GNSS constellations	GPS, GLONASS, Galileo, Beidou
Interfaces	RS-232, RS-422, CAN
Configuration	Dual-antenna / RTK
Update frequency	Up to 200 Hz
Operating temperature	−40°C to 85°C

Appendix B

Video from Field Experiments with an Encrypted Control and Guidance Systems

B.1 Video from experiments with encrypted control system

Video documenting excerpts from experiments 1 and 3 in Chapter 4 is available at:

- https://drive.google.com/file/d/1BPO52Bs1mk3ujUOCIUPl83awrbdg6SEp/view?usp=sharing

B.2 Video from experiments with encrypted guidance system

Video documenting experiments 1 and 3 in Chapter 5 are available online at:

- Experiment 1: https://drive.google.com/file/d/1r797tu1csaw0IJns1eeQFY9WWq8HBK9L/view?usp=sharing
- Experiment 3: https://drive.google.com/file/d/1EBvobEXs-ckqqKfYIEFpNNHAWVpNicRt/view?usp=sharing

References

[1] J. de Vos, R. G. Hekkenberg, and O. A. Valdez Banda, "The impact of autonomous ships on safety at sea – a statistical analysis," *Reliability Engineering & System Safety*, vol. 210, p. 107558, 2021.

[2] A. Felski and K. Zwolak, "The ocean-going autonomous ship—challenges and threats," *Journal of Marine Science and Engineering*, vol. 8, no. 1, 2020.

[3] R. R. Negenborn, F. Goerlandt, T. A. Johansen, P. Slaets, O. A. V. Banda, T. Vanelslander, and N. P. Ventikos, "Autonomous ships are on the horizon: here's what we need to know," *Nature*, vol. 615, pp. 30–33, 2023.

[4] MASS World News, "World's first uncrewed freight route at sea in the Trondheimsfjord in Norway." online, 2023. (accessed: 06/03/2023).

[5] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[6] D. Dzung, M. Naedele, T. Von Hoff, and M. Crevatin, "Security for Industrial Communication Systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1152–1177, 2005.

[7] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proceedings of the 1st International Conference on High Confidence Networked Systems*, HiCoNS '12, (New York, NY, USA), p. 55–64, Association for Computing Machinery, 2012.

[8] X. Zhang, Q. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, "Networked control systems: a survey of trends and techniques," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 1–17, 2020.

[9] A. Jadhav and S. Mutreja, "Autonomous Ships Market," *Allied Market Research - Freight & Logistics*, Dec. 2020.

[10] Research and Markets, "Global Autonomous Ships Market Report 2021-2030: Increasing Threat of Cybersecurity and Privacy is Expected to Limit Market Growth," *GlobeNewswire*, Jun. 2021.

[11] J. E. Vinnem and I. B. Utne, "Risk from cyberattacks on autonomous ships," in *Safety and Reliability – Safe Societies in a Changing World: Proceedings of ESREL 2018 June 17-21, 2018, Trondheim, Norway (1st ed.). CRC Press.*, pp. 1485–1492, 2018.

[12] Y. Xia, "Cloud control systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, pp. 134–142, Apr. 2015.

[13] R. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Processing Magazine*, vol. 30, pp. 82–105, Jan. 2013.

[14] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-Vehicle Networks: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2015.

[15] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[16] Z. El-Rewini, K. Sadatsharan, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity challenges in vehicular communications," *Vehicular Communications*, vol. 23, p. 100214, 2020.

[17] Q. Wang and H. Yang, "A survey on the recent development of securing the networked control systems," *Systems Science & Control Engineering*, vol. 7, no. 1, pp. 54–64, 2019.

[18] Y. Tan, J. Wang, J. Liu, and Y. Zhang, "Unmanned Systems Security: Models, Challenges, and Future Directions," *IEEE Network*, vol. 34, no. 4, pp. 291–297, 2020.

[19] B. Silverajan, M. Ocak, and B. Nagel, "Cybersecurity Attacks and Defences for Unmanned Smart Ships," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 15–20, 2018.

[20] V. Bolbot, G. Theotokatos, E. Boulougouris, and D. Vassalos, "A novel cyber-risk assessment method for ship systems," *Safety Science*, vol. 131, p. 104908, 2020.

[21] X. Sun, F. R. Yu, and P. Zhang, "A Survey on Cyber-Security of Connected and Autonomous Vehicles (CAVs)," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–20, 2021.

[22] G. Kavallieratos, S. Katsikas, and V. Gkioulos, "Cyber-Attacks Against the Autonomous Ship," in *Computer Security* (S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, A. Antón, S. Gritzalis, J. Mylopoulos, and

C. Kalloniatis, eds.), (Cham), pp. 20–36, Springer International Publishing, 2019.

[23] E. Yağdereli, C. Gemci, and A. Z. Aktaş, "A study on cyber-security of autonomous and unmanned vehicles," *The Journal of Defense Modeling and Simulation*, vol. 12, no. 4, pp. 369–381, 2015.

[24] T. U. Kang, H. M. Song, S. Jeong, and H. K. Kim, "Automated Reverse Engineering and Attack for CAN Using OBD-II," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–7, 2018.

[25] M. S. Lund, O. S. Hareide, and Ø. Jøsok, "An Attack on an Integrated Navigation System," *Necesse*, 2018.

[26] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, and K. Li, "A Survey of Intrusion Detection for In-Vehicle Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 919–933, 2020.

[27] M. L. Han, B. I. Kwak, and H. K. Kim, "Event-Triggered Interval-Based Anomaly Detection and Attack Identification Methods for an In-Vehicle Network," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2941–2956, 2021.

[28] K. A. Jallad, M. Aljnidi, and M. S. Desouki, "Anomaly detection optimization using big data and deep learning to reduce false-positive," *Journal of Big Data*, vol. 7, no. 68, 2020.

[29] S. Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," *ACM Transactions on Information and System Security*, vol. 3, no. 3, pp. 186–205, 2000.

[30] R. A. Gupta and M. Chow, "Performance assessment and compensation for secure networked control systems," in *2008 34th Annual Conference of IEEE Industrial Electronics*, pp. 2929–2934, 2008.

[31] Z. Pang, G. Zheng, G. Liu, and C. Luo, "Secure transmission mechanism for networked control systems under deception attacks," in *2011 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, pp. 27–32, 2011.

[32] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," tech. rep., Dept. of Elect. Eng. and Comp. Sci., Univ. of California at Berkeley, Berkeley, CA, USA, 2009.

[33] S. Bera, S. Misra, and J. J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, pp. 1477–1494, May 2015.

[34] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Network*, vol. 26, pp. 21–28, May-Jun. 2012.

[35] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, pp. 398–409, Apr. 2015.

[36] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.

[37] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pp. 162–167, 1986.

[38] W. E. Curran, C. A. Rojas, L. Bobadilla, and D. A. Shell, "Oblivious sensor fusion via secure multi-party combinatorial filter evaluation," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 5620–5627, 2021.

[39] A. B. Alexandru and G. J. Pappas, *Secure Multi-party Computation for Cloud-Based Control*, pp. 179–207. Singapore: Springer Singapore, 2020.

[40] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y. an Tan, "Secure multi-party computation: Theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, Feb. 2019.

[41] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography* (S. Halevi and T. Rabin, eds.), (Berlin, Heidelberg), pp. 265–284, Springer Berlin Heidelberg, 2006.

[42] A. Narayanan and V. Shmatikov, "How to break anonymity of the netflix prize dataset," 2006.

[43] J. Le Ny and G. J. Pappas, "Differentially private filtering," *IEEE Transactions on Automatic Control*, vol. 59, pp. 341–354, Feb. 2014.

[44] K. Yazdani, A. Jones, K. Leahy, and M. Hale, "Differentially private lq control," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 1061–1068, 2023.

[45] M. Hale, A. Jones, and K. Leahy, "Privacy in feedback: The differentially private lqg," in *2018 Annual American Control Conference (ACC)*, pp. 3386–3391, 2018.

[46] A. Moradi, N. K. D. Venkategowda, S. P. Talebi, and S. Werner, "Privacy-preserving distributed kalman filtering," *IEEE Transactions on Signal Processing*, vol. 70, pp. 3074–3089, Jun. 2022.

[47] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *Applied Cryptography and Network Security* (M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, eds.), (Berlin, Heidelberg), pp. 102–118, Springer Berlin Heidelberg, 2013.

[48] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: Machine learning on encrypted data," in *Information Security and Cryptology – ICISC 2012* (T. Kwon, M.-K. Lee, and D. Kwon, eds.), (Berlin, Heidelberg), pp. 1–21, Springer Berlin Heidelberg, 2013.

[49] J. D. Cohen and M. J. Fischer, "A robust and verifiable cryptographically secure election scheme," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pp. 372–382, 1985.

[50] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, Feb. 1978.

[51] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proceedings of CRYPTO 84 on Advances in Cryptology*, (Santa Barbara, CA, USA), pp. 10–18, Aug. 19-22 1984.

[52] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*, (Prague, Czech Republic), pp. 223–238, May 2-6 1999.

[53] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of Cryptography* (J. Kilian, ed.), (Berlin, Heidelberg), pp. 325–341, Springer Berlin Heidelberg, 2005.

[54] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, (New York, NY, USA), p. 169–178, Association for Computing Machinery, 2009.

[55] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022.

[56] K. Kogiso and T. Fujita, "Cyber-security enhancement of networked control systems using homomorphic encryption," in *2015 54th IEEE Conference on Decision and Control (CDC)*, (Osaka, Japan), pp. 6836–6843, Dec. 15-18 2015.

[57] M. Schulze Darup, A. B. Alexandru, D. E. Quevedo, and G. J. Pappas, "Encrypted control for networked systems: An illustrative introduction and current challenges," *IEEE Control Systems Magazine*, vol. 41, pp. 58–78, Jun. 2021.

[58] F. Farokhi, I. Shames, and N. Batterham, "Secure and private cloud-based control using semi-homomorphic encryption," *IFAC-PapersOnLine*, vol. 49, pp. 163–168, Sep. 2016.

[59] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cloud-based mpc for linear systems with input constraints," *IFAC-PapersOnLine*, vol. 51, pp. 535–542, Aug. 2018.

[60] M. Schulze Darup, A. Redder, I. Shames, F. Farokhi, and D. Quevedo, "Towards encrypted mpc for linear constrained systems," *IEEE Control Systems Letters*, vol. 2, pp. 195–200, Apr. 2018.

[61] A. B. Alexandru, M. Morari, and G. J. Pappas, "Cloud-based mpc with encrypted data," in *2018 IEEE Conference on Decision and Control (CDC)*, (Miami Beach, FL, USA), pp. 5014–5019, Dec. 17-19 2018.

[62] M. Schulze Darup, A. Redder, and D. E. Quevedo, "Encrypted cooperative control based on structured feedback," *IEEE Control Systems Letters*, vol. 3, pp. 37–42, Jan. 2019.

[63] A. B. Alexandru, M. Schulze Darup, and G. J. Pappas, "Encrypted cooperative control revisited," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, (Nice, France), pp. 7196–7202, Dec. 11-13 2019.

[64] M. Aristov, B. Noack, U. D. Hanebeck, and J. Müller-Quade, "Encrypted multisensor information filtering," in *2018 21st International Conference on Information Fusion (FUSION)*, pp. 1631–1637, 2018.

[65] A. B. Alexandru and G. J. Pappas, "Encrypted LQG using labeled homomorphic encryption," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, (Montreal, Quebec, Canada), pp. 129–140, Apr. 16–18 2019.

[66] Z. Zhang, P. Cheng, J. Wu, and J. Chen, "Secure state estimation using hybrid homomorphic encryption scheme," *IEEE Transactions on Control Systems Technology*, vol. 29, pp. 1704–1720, Jul. 2021.

[67] M. Ristic, B. Noack, and U. D. Hanebeck, "Secure fast covariance intersection using partially homomorphic and order revealing encryption schemes," *IEEE Control Systems Letters*, vol. 5, pp. 217–222, Jan. 2021.

[68] M. Ristic and B. Noack, "Encrypted fast covariance intersection without leaking fusion weights," in *2022 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–6, 2022.

[69] N. Schlüter and M. S. Darup, "On the stability of linear dynamic controllers with integer coefficients," *IEEE Transactions on Automatic Control*, vol. 67, pp. 5610–5613, Oct. 2022.

[70] C. Murguia, F. Farokhi, and I. Shames, "Secure and private implementation of dynamic controllers using semihomomorphic encryption," *IEEE Transactions on Automatic Control*, vol. 65, no. 9, pp. 3950–3957, 2020.

[71] J. Tran, F. Farokhi, M. Cantoni, and I. Shames, "Implementing homomorphic encryption based secure feedback control," *Control Engineering Practice*, vol. 97, pp. 104350–104362, Apr. 2020.

[72] F. Farokhi, I. Shames, and N. Batterham, "Secure and private control using semi-homomorphic encryption," *Control Engineering Practice*, vol. 67, pp. 13–20, Oct. 2017.

[73] J. H. Cheon, K. Han, S.-M. Hong, H. J. Kim, J. Kim, S. Kim, H. Seo, H. Shim, and Y. Song, "Toward a secure drone system: Flying with real-time homomorphic authenticated encryption," *IEEE Access*, vol. 6, pp. 24325–24339, Mar. 2018.

[74] Ø. Volden, P. Solnør, S. Petrovic, and T. I. Fossen, "Secure and Efficient Transmission of Vision-Based Feedback Control Signals," *Journal of Intelligent & Robotic Systems*, vol. 103, pp. 1–16, Oct. 2021.

[75] P. Solnør, Ø. Volden, K. Gryte, S. Petrovic, and T. I. Fossen, "Hijacking of unmanned surface vehicles: A demonstration of attacks and countermeasures in the field," *Journal of Field Robotics*, vol. 39, pp. 631–649, Aug. 2022.

[76] P. Solnør, S. Petrovic, and T. I. Fossen, "Development and experimental validation of an encrypted control system for an unmanned surface vehicle," *Robotics and Autonomous Systems*, pp. 1–17, 2023. (Submitted).

[77] P. Solnør, S. Petrovic, and T. I. Fossen, "Towards oblivious guidance systems for autonomous vehicles," *IEEE Transactions on Vehicular Technology*, pp. 1–15, 2023. (Accepted, in press).

[78] P. Solnør and A. G. Hem, "Revisiting encrypted fast covariance intersection," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–11, 2023. (Submitted).

[79] P. Solnør, "A Cryptographic Toolbox for Feedback Control Systems," *Modeling, Identification and Control*, vol. 41, pp. 313–332, Dec. 2020.

[80] M. Akdağ, P. Solnør, and T. A. Johansen, "Collaborative collision avoidance for maritime autonomous surface ships: A review," *Ocean Engineering*, vol. 250, p. 110920, Apr. 2022.

[81] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. Hoboken, NJ, USA: Wiley, 2nd ed., 2021.

[82] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3d lidar and camera data for object detection in autonomous vehicle applications," *IEEE Sensors Journal*, vol. 20, no. 9, pp. 4901–4913, 2020.

[83] H. Song, K. Lee, and D. H. Kim, "Obstacle avoidance system with lidar sensor based fuzzy control for an autonomous unmanned ship," in *2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS)*, pp. 718–722, 2018.

[84] P. Trslic, M. Rossi, L. Robinson, C. W. O'Donnel, A. Weir, J. Coleman, J. Riordan, E. Omerdic, G. Dooly, and D. Toal, "Vision based autonomous docking for work class rovs," *Ocean Engineering*, vol. 196, p. 106840, 2020.

[85] K. Ji and W.-j. Kim, "Real-time control of networked control systems via ethernet," *International Journal of Control Automation and Systems*, vol. 3, 11 2004.

[86] National Bureau of Standards, "Data Encryption Standard (DES)." Federal Information Processing Standards Publication 46, 1977.

[87] E. Barker and N. Mouha, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher," tech. rep., 2017-11-17 2017.

[88] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)." Federal Information Processing Standards Publication 197, 2001.

[89] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, RFC Editor, 4 1992.

[90] Q. H. Dang, "The Keyed-Hash Message Authentication Code (HMAC) - FIPS 198-1," tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, USA, 2008.

[91] Z. Pang and G. Liu, "Design and implementation of secure networked predictive control systems under deception attacks," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1334–1342, 2012.

[92] J. Jithish and S. Sankaran, "Securing networked control systems: Modeling attacks and defenses," in *2017 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 7–11, 2017.

[93] J. H. An and M. Bellare, "Does encryption with redundancy provide authenticity?," in *Advances in Cryptology — EUROCRYPT 2001* (B. Pfitzmann, ed.), (Berlin, Heidelberg), pp. 512–528, Springer Berlin Heidelberg, 2001.

[94] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," *J. Cryptol.*, vol. 21, p. 469–491, Sept. 2008.

[95] R. R. Teixeira, I. P. Maurell, and P. L. J. Drews, "Security on ROS: analyzing and exploiting vulnerabilities of ROS-based systems," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pp. 1–6, 2020.

[96] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *International Workshop on Fast Software Encryption*, pp. 191–204, Springer, 1993.

[97] F. J. R. Lera, J. Balsa, F. Casado, C. Fernández, F. M. Rico, and V. Matellán, "Cybersecurity in autonomous systems: Evaluating the performance of hardening ROS," *Málaga, Spain*, vol. 47, 2016.

[98] F. J. Rodríguez-Lera, V. Matellán-Olivera, J. Balsa-Comerón, Á. M. Guerrero-Higueras, and C. Fernández-Llamas, "Message encryption in robot operating system: Collateral effects of hardening mobile robots," *Frontiers in ICT*, vol. 5, p. 11, 2018.

[99] J. Balsa-Comerón, Á. M. Guerrero-Higueras, F. J. Rodríguez-Lera, C. Fernández-Llamas, and V. Matellán-Olivera, "Cybersecurity in autonomous systems: Hardening ROS using encrypted communications and semantic rules," in *ROBOT 2017: Third Iberian Robotics Conference* (A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, eds.), (Cham), pp. 67–78, Springer International Publishing, 2018.

[100] M. Robshaw, *The eSTREAM Project*, pp. 1–6. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[101] H. Wu and B. Preneel, "AEGIS: A Fast Authenticated Encryption Algorithm," in *Selected Areas in Cryptography – SAC 2013* (T. Lange, K. Lauter, and P. Lisoněk, eds.), (Berlin, Heidelberg), pp. 185–201, Springer Berlin Heidelberg, 2014.

[102] Ø. Volden and P. Solnør, "Crypto ROS: Secure and Efficient Transmission of Vision-Based Feedback Control Signals." https://github.com/oysteinvolden/Real-time-sensor-encryption, 2020.

[103] H. Wu, *The Stream Cipher HC-128*, p. 39–47. Berlin, Heidelberg: Springer-Verlag, 2008.

[104] D. Bernstein, "ChaCha, a variant of Salsa20," 01 2008.

[105] M. Boesgaard, M. Vesterager, and E. Zenner, *The Rabbit Stream Cipher*, p. 69–83. Berlin, Heidelberg: Springer-Verlag, 2008.

[106] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, *Sosemanuk, a fast software-oriented stream cipher*, p. 98–118. Berlin, Heidelberg: Springer-Verlag, 2008.

[107] Quynh H. Dang, "Secure Hash Standard - Federal Information Processing Standard Publication 180-4," tech. rep., National Institute of Standards and Technology, Gaithersburg, MD, USA, 2015.

[108] S. Turner and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 algorithms," RFC 6151, 3 2011.

[109] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 3, pp. 365–403, 2003.

[110] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[111] e. a. T. Boutell, "RFC 2083: PNG (Portable Network Graphics) Specification," March 1997.

[112] H. ZainEldin, M. A. Elhosseini, and H. A. Ali, "Image compression algorithms in wireless multimedia sensor networks: A survey," *Ain Shams Engineering Journal*, vol. 6, no. 2, pp. 481–490, 2015.

[113] "Image file reading and writing." https://docs.opencv.org/3.4.3/d4/da8/group__imgcodecs.html, 2018. Accessed: 2020-06-20.

[114] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[115] Rolls-Royce, "Rolls-Royce and Finferries demonstrate world's first Fully Autonomous Ferry," *Rolls-Royce Press Release*, 12 2018.

[116] L. Quinton, "Wärtsilä to develop autonomous, zero emission barge for port of rotterdam," *Wärtsilä Corporation Press Release*, 5 2021.

[117] NTNU, "Autonomous all-electric passenger ferries for urban water transport (autoferry)." https://www.ntnu.edu/autoferry, 2021. Accessed: 2021-09-29.

[118] J. Pinto, P. S. Dias, R. Martins, J. Fortuna, E. Marques, and J. Sousa, "The LSTS toolchain for networked vehicle systems," in *2013 MTS/IEEE OCEANS - Bergen*, pp. 1–9, 2013.

[119] ABI Research, "The Rise of ROS: Nearly 55% of total commercial robots shipped in 2024 will have at least one Robot Operating System package installed," *Business Wire*, 09 2019.

[120] B. Dieber, R. White, S. Taurer, B. Breiling, G. Caiazza, H. Christensen, and A. Cortesi, "Penetration Testing ROS," in *Robot Operating System (ROS): The Complete Reference (Volume 4)* (A. Koubaa, ed.), (Cham), pp. 183–225, Springer International Publishing, 2020.

[121] K. Fazzari, "ROS 2 DDS-Security integration," 2021. https://design.ros2.org/articles/ros2_dds_security, Accessed: 2021-09-29.

[122] H. Mun, K. Han, and D. H. Lee, "Ensuring Safety and Security in CAN-Based Automotive Embedded Systems: A Combination of Design Optimization and Secure Communication," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7078–7091, 2020.

[123] E. Barker and A. Roginsky, "NIST Special Publication 800-131A - Transitioning the Use of Cryptographic Algorithms and Key Lengths, Revision 2," *NIST Technical Series Publications*, 03 2019.

[124] D. R. Stinson and M. Paterson, *Cryptography: Theory and Practice, Fourth Edition*. Boca Raton: Chapman and Hall/CRC, 2018.

[125] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control. 1st Edition*. Wiley, 2011.

[126] E. Borhaug, A. Pavlov, and K. Y. Pettersen, "Integral los control for path following of underactuated marine surface vessels in the presence of constant ocean currents," in *2008 47th IEEE Conference on Decision and Control*, pp. 4984–4991, 2008.

[127] M. Fox, "NetfilerQueue," 2021. https://pypi.org/project/NetfilterQueue/, Accessed: 2021-10-25.

[128] P. Biondi, "Scapy," 2021. https://scapy.net, Accessed: 2021-10-22.

[129] Analog Devices, "ADIS16490," 2021. https://www.analog.com/media/en/technical-documentation/data-sheets/adis16490.pdf, Accessed: 2021-10-26.

[130] U-blox, "ZED-F9P module," 2021. https://www.u-blox.com/en/product/zed-f9p-module, Accessed: 2021-10-22.

[131] SentiSystems, "SentiPack," 2021. https://sentisolution.com/wp-content/uploads/2020/08/datasheet.pdf, Accessed: 2021-10-26.

[132] SBG Systems, "Ellipse Series," 2021. https://www.sbg-systems.com/products/ellipse-series/#ellipse-d_rtk_gnss_ins, Accessed: 2021-10-22.

[133] W. Caharija, M. Candeloro, K. Y. Pettersen, and A. J. Sørensen, "Relative Velocity Control and Integral LOS for Path Following of Underactuated Surface Vessels," *IFAC Proceedings Volumes*, vol. 45, no. 27, pp. 380–385, 2012. 9th IFAC Conference on Manoeuvring and Control of Marine Craft.

[134] W. Caharija, *Integral Line-of-Sight Guidance and Control of Underactuated Marine Vehicles*. PhD thesis, Norwegian University of Science and Technology, 2014.

[135] Z. Chaloupka, N. Alsindi, and J. Aweya, "Transparent clock characterization using IEEE 1588 PTP timestamping probe," in *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pp. 1537–1542, 2015.

[136] J. Kridner, G. Coley, and R. P. Day, "Beaglebone black system reference manual," 2021. https://github.com/beagleboard/beaglebone-black/wiki/System-Reference-Manual, Accessed: 2021-10-26.

[137] Nvidia, "Jetson AGX Xavier Developer Kit," 2021. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit, Accessed: 2021-10-26.

[138] S. M. Albrektsen, *Sensor Synchronization and Navigation in GNSS-Denied Environments for Unmanned Aerial Vehicles*. PhD thesis, Norwegian University of Science and Technology, 2018.

[139] T. Abdelzaher, Y. Hao, K. Jayarajah, A. Misra, P. Skarin, S. Yao, D. Weerakoon, and K.-E. Årzén, "Five challenges in cloud-enabled intelligence and control," *ACM Trans. Internet Technol.*, vol. 20, pp. 1–19, Feb. 2020.

[140] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite, "Control-as-a-service from the cloud: A case study for using virtualized plcs," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, (Toulouse, France), pp. 1–4, May 5-7 2014.

[141] H. Esen, M. Adachi, D. Bernardini, A. Bemporad, D. Rost, and J. Knodel, "Control as a service (caas): Cloud-based software architecture for automotive control applications," in *Proceedings of the Second International Workshop on the Swarm at the Edge of the Cloud*, SWEC '15, (New York, NY, USA), p. 13–18, Association for Computing Machinery, 2015.

[142] A. Vick, V. Vonásek, R. Pěnička, and J. Krüger, "Robot control as a service — towards cloud-based motion planning and control for industrial robots," in *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, (Poznan, Poland), pp. 33–39, Jul. 6-8 2015.

[143] A. Teixeira, K. C. Sou, H. Sandberg, and K. H. Johansson, "Secure control systems: A quantitative risk management approach," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 24–45, 2015.

[144] K. Kogiso, R. Baba, and M. Kusaka, "Development and examination of encrypted control systems," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, (Auckland, New Zealand), pp. 1338–1343, Jul. 9-12 2018.

[145] K. Teranishi and K. Kogiso, "Elgamal-type encryption for optimal dynamic quantizer in encrypted control systems," *SICE Journal of Control, Measurement, and System Integration*, vol. 14, pp. 59–66, Apr. 2021.

[146] K. Teranishi and K. Kogiso, "Encrypted gain scheduling with quantizers for stability guarantee," in *2021 60th IEEE Conference on Decision and Control (CDC)*, (Austin, TX, USA), pp. 5628–5633, Dec. 13-17 2021.

[147] M. Barbosa, D. Catalano, and D. Fiore, "Labeled homomorphic encryption: Scalable and privacy-preserving processing of outsourced data." Cryptology ePrint Archive, Report 2017/326, 2017. https://ia.cr/2017/326.

[148] M. S. Darup and T. Jager, "Encrypted cloud-based control using secret sharing with one-time pads," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 7215–7221, 2019.

[149] M. Schulze Darup, "Encrypted polynomial control based on tailored two-party computation," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 11, pp. 4168–4187, 2020.

[150] J. Kim, C. Lee, H. Shim, J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Encrypting controller using fully homomorphic encryption for security of cyber-physical systems," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 175 – 180, 2016. 6th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2016.

[151] K. Teranishi, N. Shimada, and K. Kogiso, "Development and examination of fog computing-based encrypted control system," *IEEE Robotics and Automation Letters*, vol. 5, pp. 4642–4648, Jul. 2020.

[152] D. R. Stinson and M. B. Paterson, *Cryptography - Theory and Practice*. Boca Raton, FL, USA: CRC Press, Inc., 4th ed., 2019.

[153] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[154] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, (San Francisco, CA, USA), pp. 365–377, May 5-7 1982.

[155] M. Joye and B. Libert, "Efficient cryptosystems from 2k-th power residue symbols," in *Advances in Cryptology – EUROCRYPT 2013*, (Athens, Greece), pp. 76–92, May 26-30 2013.

[156] F. Benhamouda, J. Herranz, M. Joye, and B. Libert, "Efficient cryptosystems from 2^k-th power residue symbols." Cryptology ePrint Archive, Report 2013/435, 2013. https://ia.cr/2013/435.

[157] K. Teranishi and K. Kogiso, "Dynamic quantizer for encrypted observer-based control," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 5477–5482, 2020.

[158] T. Samad, "A survey on industry impact and challenges thereof [technical activities]," *IEEE Control Systems Magazine*, vol. 37, no. 1, pp. 17–18, 2017.

[159] P. Solnør, "Labeled homomorphic control." https://github.com/pettsol/LabeledHomomorphicControl, 2022.

[160] E. Barkin, "Sp 800-57 revision 5. recommendation for key management," Tech. Rep. SP 800-57, National Institute of Standards and Technology, Gaithersburg, MD, USA, May 2020.

[161] National Institute of Standards and Technology, "Sha-3 standard: Permutation-based hash and extendable-output functions - fips 202," tech. rep., Gaithersburg, MD, USA, 2015.

[162] T. Granlund and the GMP development team, *GNU MP: The GNU Multiple Precision Arithmetic Library*, 6.2.1 ed., 2022. https://gmplib.org/gmp-man-6.2.1.pdf.

[163] X. Zhao, Z. Cao, X. Dong, J. Shao, L. Wang, and Z. Liu, *New Assumptions and Efficient Cryptosystems from the e-th Power Residue Symbol*, pp. 408–424. 08 2020.

[164] Y. Xia, Y. Zhang, L. Dai, Y. Zhan, and Z. Guo, "A brief survey on recent advances in cloud control systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, pp. 3108–3114, Jul. 2022.

[165] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 7944–7956, Aug. 2019.

[166] C. Gao, G. Wang, W. Shi, Z. Wang, and Y. Chen, "Autonomous driving security: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 9, pp. 7572–7595, May 2022.

[167] Z. Liu, J. Ma, J. Weng, F. Huang, Y. Wu, L. Wei, and Y. Li, "Lppte: A lightweight privacy-preserving trust evaluation scheme for facilitating distributed data fusion in cooperative vehicular safety applications," *Information Fusion*, vol. 73, pp. 144–156, Sep. 2021.

[168] Z. Liu, J. Weng, J. Guo, J. Ma, F. Huang, H. Sun, and Y. Cheng, "Pptm: A privacy-preserving trust management scheme for emergency message dissemination in space–air–ground-integrated vehicular networks," *IEEE Internet of Things Journal*, vol. 9, pp. 5943–5956, Apr. 2022.

[169] Y. Cheng, J. Ma, Z. Liu, Y. Wu, K. Wei, and C. Dong, "A lightweight privacy preservation scheme with efficient reputation management for mobile crowdsensing in vehicular networks," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[170] F. Farivar, M. Sayad Haghighi, A. Jolfaei, and S. Wen, "On the security of networked control systems in smart vehicle and its adaptive cruise control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 3824–3831, Jun. 2021.

[171] A. M. Naseri, W. Lucia, and A. Youssef, "Encrypted cloud-based set-theoretic model predictive control," *IEEE Control Systems Letters*, vol. 6, pp. 3032–3037, Jun. 2022.

[172] K. Teranishi, T. Sadamoto, A. Chakrabortty, and K. Kogiso, "Designing optimal key lengths and control laws for encrypted control systems based on sample identifying complexity and deciphering time," *IEEE Transactions on Automatic Control*, 2022.

[173] T. I. Fossen and K. Y. Pettersen, "On uniform semiglobal exponential stability (usges) of proportional line-of-sight guidance laws," *Automatica*, vol. 50, pp. 2912–2917, Nov. 2014.

[174] H. K. Khalil, *Nonlinear systems*. Englewood Cliffs, NJ, USA: Patience Hall, 3rd ed., 2002.

[175] P. Solnør, "Encrypted guidance." https://github.com/pettsol/Encrypted-Guidance, 2022.

[176] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, Mar. 2016.

[177] P. Arthurs, L. Gillam, P. Krause, N. Wang, K. Halder, and A. Mouzakitis, "A taxonomy and survey of edge cloud computing for intelligent transportation systems and connected vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 6206–6221, Jul. 2022.

[178] M. Ahmadian and D. C. Marinescu, "Information leakage in cloud data warehouses," *IEEE Transactions on Sustainable Computing*, vol. 5, pp. 192–203, Apr.-Jun. 2020.

[179] R. Singer and A. Kanyuck, "Computer control of multiple site track correlation," *Automatica*, vol. 7, pp. 455–463, Jul. 1971.

[180] S. Matzka and R. Altendorfer, "A comparison of track-to-track fusion algorithms for automotive sensor fusion," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 189–194, Aug 2008.

[181] S. Coraluppi and C. Carthel, "Recursive track fusion for multi-sensor surveillance," *Information Fusion*, vol. 5, pp. 23 – 33, Mar. 2004.

[182] Y. Bar-Shalom, "On the track-to-track correlation problem," *IEEE Transactions on Automatic Control*, vol. 26, pp. 571–572, Apr. 1981.

[183] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT, USA: YBS Publishing, 1995.

[184] J. Uhlmann, *Dynamic Map Building and Localization: New Theoretical Foundations*. PhD thesis, University of Oxford, 1995.

[185] W. Niehsen, "Information fusion based on fast covariance intersection filtering," in *Proceedings of the Fifth International Conference on Information Fusion. FUSION 2002. (IEEE Cat.No.02EX5997)*, vol. 2, pp. 901–904 vol.2, 2002.

[186] C. Castelluccia, A. C.-F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, pp. 1–36, Jun. 2009.

[187] X. Yan, B. Chen, Y. Zhang, and L. Yu, "Guaranteeing differential privacy in distributed fusion estimation," *IEEE Transactions on Aerospace and Electronic Systems*, pp. 1–13, 2022.

[188] J. Wang, J.-F. Zhang, and X.-K. Liu, "Differentially private resilient distributed cooperative online estimation over digraphs," *International Journal of Robust and Nonlinear Control*, vol. 32, pp. 8670–8688, Oct. 2022.

[189] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 201–210, JMLR.org, 2016.

[190] S. Julier and J. K. Uhlmann, "General decentralized data fusion with covariance intersection (ci)," in *Handbook of Data Fusion* (D. Hall and J. Llinas, eds.), Boca Raton, FL, USA: CRC Press LLC, 2001.

[191] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Application to Tracking and Navigation*. Hoboken, NJ, USA: Wiley, 2001.

[192] E. Shi, T. H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*, (Reston, VA, USA), The Internet Society, 2011.

[193] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data," in *Financial Cryptography and Data Security* (A.-R. Sadeghi, ed.), (Berlin, Heidelberg), pp. 111–125, Springer Berlin Heidelberg, 2013.

[194] F. Benhamouda, M. Joye, and B. Libert, "A new framework for privacy-preserving aggregation of time-series data," *ACM Trans. Inf. Syst. Secur.*, vol. 18, pp. 1–21, Apr. 2016.

[195] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," *Dokl. Akad. Nauk SSSR*, vol. 145, pp. 293–294, Feb. 1962.

[196] A. L. Toom, "The complexity of a scheme of functional elements simulating the multiplication of integers," *Dokl. Akad. Nauk SSSR*, vol. 150, pp. 496–498, Jan. 1963.

[197] S. Cook, *On the Minimum Computation Time of Functions*. PhD thesis, Harvard University, 1966.

[198] P. Solnør, "Revisiting encrypted fast covariance intersection." https://github.com/pettsol/Revisiting-encrypted-FCI, 2022.

[199] N. Bore, "IMC - ROS Bridge: A Minimal library for bridging ROS and IMC messages." https://github.com/smarc-project/imc_ros_bridge, 2021.