applied sciences

MDPI

Article

Security Architecture for Swarms of Autonomous Vehicles in Smart Farming

Belén Martínez-Rodríguez *, Sonia Bilbao-Arechabala and Fernando Jorge-Hernandez

Tecnalia, Basque Research and Technology Alliance (BRTA), Parque Científico y Tecnológico de Bizkaia, C/Geldo, Edificio 700, 48160 Derio, Spain; sonia.bilbao@tecnalia.com (S.B.-A.); fernando.jorge@tecnalia.com (F.J.-H.)
* Correspondence: belen.martinez@tecnalia.com

Featured Application: The security architecture proposed can be implemented in any application that involves the use of autonomous vehicles, both terrestrial and aerial, e.g., monitoring of crops and livestock behaviour with UAVs (Unmanned Aerial Vehicles); autonomous control of implements in tractors to prepare the ground, to sow the seeds, to apply precise amounts of fertilizer or to maintain the crops; surveillance of areas with UAVs to detect intruders or wild animals; UAV-based sampling systems; etc.

Abstract: Nowadays, autonomous vehicles are incorporated into farms to facilitate manual labour. Being connected vehicles, as IoT systems, they are susceptible to cyber security attacks that try to cause damage to hardware, software or even living beings. Therefore, it is important to provide sufficient security mechanisms to protect both the communications and the data, mitigating any possible risk or harm to farmers, livestock or crops. Technology providers are aware of the importance of ensuring security, and more and more secure solutions can be found on the market today. However, generally, these particular solutions are not sufficient when they are part of complex hybrid systems, since there is no single global solution proposal. In addition, as the number of technologies and protocols used increases, the number of security threats also increases. This article presents a cybersecurity architecture proposal for swarms of heterogeneous vehicles in smart farming, which covers all of the aspects recommended by the ISO 7798-2 specification in terms of security. As a result of this analysis, a detailed summary of the possible solutions and available technologies for each of the communication channels of the target system as well as some recommendations are presented.

Keywords: security; autonomous vehicles; smart farming

Citation: Martínez-Rodríguez, B.; Bilbao-Arechabala, S.; Jorge-Hernandez, F. Security Architecture for Swarms of Autonomous Vehicles in Smart Farming. *Appl. Sci.* **2021**, *11*, 4341. https://doi.org/10.3390/app11104341

Academic Editor: Yosoon Choi

Received: 25 March 2021
Accepted: 8 May 2021
Published: 11 May 2021

1. Introduction

The global population is constantly growing. In 2019, the United Nations estimated that the world's population is expected to increase by 2 billion persons in the next 30 years, reaching 9.7 billion in 2050 and potentially reaching 11 billion around 2100 [1]. Many of the fastest growing populations are in the poorest countries, which presents threats to sustainable development [2]. Moreover, by 2050, it is projected that 68 percent of the world's population will live in urban areas (an increase from 54 percent in 2016) [3].

Due to labour shortage, depopulation of rural areas and increases in food demand among others, farming faces many challenges in terms of productivity and cost-effectiveness. The usage of autonomous vehicles in agriculture is the current means to overcome these limitations while fostering new opportunities for the farming industry to bring higher yields.

Drones and autonomous tractors have started to be used in experimental farms to till, plant and harvest crops, as they can operate 24/7 [4,5]. GPS (Global Positioning System) tracking and navigation systems, robotics, AI (Artificial Intelligence) and computer vision techniques can make precision farming possible and enable farmers to make informed decisions. While these implementations control tractors to prepare the ground, to sow the

seeds, to apply precise amount of fertilizer and to maintain the crops, drones can survey the area to take samples and to analyse images to detect weeds or diseases. Drones can also fly to monitor the behaviour of livestock in a nonintrusive manner, e.g., when they are in heat or about to give birth, or even to look for lost cattle.

1.1. Challenges of Autonomous Vehicles

Autonomy in addition to efficiency and productivity can also bring drawbacks and a sense of unreliability. Farmers are sometimes reluctant to adopt these new technologies in fear of vehicle accidents or cyber security attacks [6,7]. How do we elude malicious use of autonomous vehicles? How can we guarantee that unauthorised users do not connect to our network and send commands to a drone or ground vehicle that can put our system in danger? How can we avoid alterations in the content of the messages? It is critical to analyse the vulnerabilities of the system and to provide a security design architecture that considers confidentiality and integrity of the data and messages, authentication and authorisation of the users or systems, and non-repudiation of data [8].

Most of the vulnerabilities mentioned above can be avoided by using a secure communication system which, in turn, depends on the technology selected for data exchange. In the case of autonomous vehicles, additional aspects need to be considered when deciding upon the data transmission technology to be used, such as multi-node data exchange, real-time constraints, reliability, dynamic discovery, message durability, fast algorithms or energy management. In this paper, the authors present DDS (Data Distribution Service) as the most appropriate communication technology for distributed critical systems.

Additionally, the maximum potential in use cases involving fleets of vehicles can be obtained when combining vehicles of different typology (aerial and terrestrial), with different capabilities and equipment (e.g., advanced sensors) and from different vendors. For example, remote sensing (information gathered by means of UAVs) and ground sensing (information collected from UGVs-Unmanned Ground Vehicles-or tractors), wide maps of environment provided by UAVs, can be used to plan precise missions for UGVs to reduce trip costs and the mission's duration. In these cases, semantic interoperability challenges need to also be addressed. Moreover, vendor-specific security solutions are no longer valid when heterogeneous ecosystems are deployed, and a global security solution is needed.

This paper addresses the research challenge of implementing a global and economical solution that covers all five areas of security (i.e., confidentiality and integrity of the data and messages, authentication and authorisation of the users or systems, and non-repudiation of data) in a multi-node communication system that guarantees the constraints of heterogeneous autonomous robots in terms of real-time, reliability, energy and QoS management.

1.2. Paper Contributions and Structure

This paper provides a security assessment, recommendations and solutions to design and implement a secure architecture for systems that manages missions with swarms of heterogeneous autonomous or unmanned vehicles in Smart Farming. In the case of heterogeneous swarms, the diverse nature of the hardware implies that the use of the particular security solutions that manufacturers may offer is not sufficient, and it is necessary to design a holistic solution able to protect the global architecture. Moreover, the security solution must not affect the performance of the vehicles, which are capable of autonomous navigation and must transmit their feedback to the control station in real time. The energy consumption of vehicles must also be taken into account, considering the limited autonomy that most autonomous commercial vehicles usually have (e.g., standard battery-powered drones typically have short endurance, 15 to 90 minutes maximum, due to battery weight limitations [9]).

The architecture and technologies proposed for managing real-time communications among autonomous vehicles is the result of the research being carried out in the project

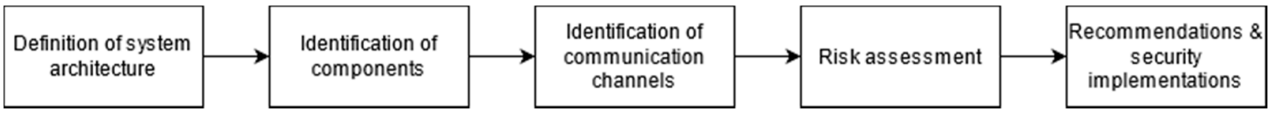
H2020-ECSEL-AFARCLOUD. This architecture and some of the security recommendations have been tested in 11 site scenarios [10].

The structure of the paper is as follows. Section 1 introduces the challenges of the research. Next, in Section 2, a communication architecture is proposed for the management of missions involving autonomous vehicles. Moreover, the state-of-the-art of security technologies is presented, related to this type of communication architecture. Then, Section 3 explains the results of the research covering all of the security implementations and recommendations. Following, Section 4 discusses the context in which these results are applicable and most valuable. Finally, Section 5 concludes the paper with a detailed summary of the main findings and cybersecurity recommendations.

2. Materials and Methods

2.1. Methodology

The methodology used in this paper to design the security architecture is an adaptation of the standard IEC 62,443 and covers the mandatory steps of a security assessment process.



The first step is the definition of the system architecture. From this definition, the core components (step 2) as well as the communication channels among all of them (step 3) are identified. In step 4, a risk assessment is carried out covering all five areas of security (i.e., confidentiality and integrity of the data and messages, authentication and authorisation of the users or systems, and non-repudiation of data). Finally, in step 5, recommendations and guidelines are provided for the security implementation.

2.2. State of the Art

2.2.1. Security Architecture Goals

Security architecture refers to the processes and tools used to prevent or mitigate attacks on a particular system. The primary purpose of the security architecture presented by the authors of this paper is to protect from cyber harm, solutions or systems for managing heterogeneous swarms of autonomous vehicles. This design is based on the security architecture of the OSI Reference Model (ISO 7798-2), which covers secure communications between open systems. This part of ISO 7798 considers that, to maintain the security of transmissions in a communication network, the following aspects must be guaranteed:

- Confidentiality, to protect the data against non-authorised revelations, ensuring that only authorised people are allowed to access the information;
- Integrity, to protect the data against non-authorised modifications, insertions or deletions;
- Authentication, to verify the supposed identity of a user or a system;
- Access control, to protect the system resources against non-authorised users; and
- Non-repudiation, to prevent an entity from denying or refuting previous commitments, responsibilities or actions.

Furthermore, correct system operation (security of process) must also be ensured by implementing measures that guarantee system recovery and backup in case of software failure, incorrect configuration or damage by malware.

The security architecture presented in this paper covers all of these aspects related to communication security, proposing and evaluating different alternatives for each of the aforementioned aspects and giving recommendations on how to apply them in a practical way to the target system depicted in Figure 1. This security architecture is described in detail in Section 3.2.

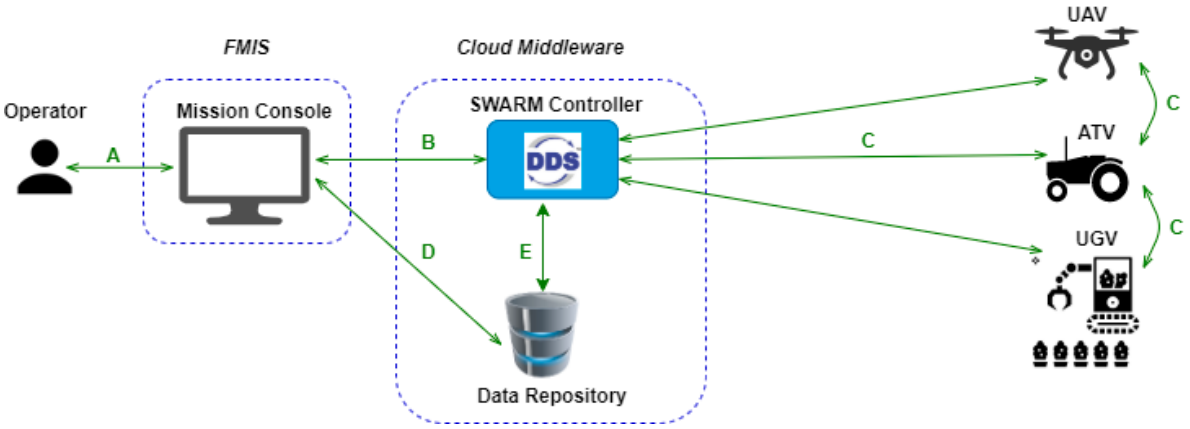


Figure 1. Core components of a system to manage missions with a swarm of autonomous vehicles.

2.2.2. Challenges in the Management of Swarms of Heterogeneous Autonomous Vehicles

Designing a solution to orchestrate a swarm of heterogeneous vehicles from different vendors is a complex task, mainly due to the lack of off-the-shelf products. Sockets was one of the first solutions used to communicate complex systems, as they were fixed with static IP addresses. However, as complexity increases, it becomes very hard to develop and maintain a socket-based solution because programmers need to manage all of the point-to-point communications. Additionally, in mobility, IP addresses can change as objects move around, so socket-based solutions are not suitable. For this reason, publish/subscribe solutions with a central broker were introduced [11]. In this case, all messages are sent to a central broker and the broker ensures that they reach the right destination. This approach avoids the need for static IPs but introduces a single point of failure because the service is interrupted if the broker fails. Moreover, as all messages go through the central broker, the performance can be slower. To overcome these limitations, other publish/subscribe technologies such as the DDS for real-time systems standard follow a distributed approach where the responsibility of the central broker is split out to all of the various nodes that make up the DDS network, enabling direct peer-to-peer communication without the need for a central message manager. In addition, DDS manages time and space decoupling, allowing late joiners to receive previous messages. Other interesting features of DDS are high-performance, real-time guarantee, scalable data exchanges and QoS policies, all of which are highly suitable for complex systems such as autonomous vehicles. In fact, DDS technology is used as a real-time communication backbone for many military unmanned vehicles [12], European Space Agency [13] and NASA robots [14,15].

The use of DDS as a real-time communication middleware has proven to be a valid solution for this type of system since DDS offers mechanisms that simplify the communication process between multiple diverse robots and their command and control systems.

2.2.3. Security Options for Swarms of Heterogeneous Autonomous Vehicles

Establishing a cyber-security architecture for autonomous vehicles is highly recommendable, as vulnerabilities in autonomous vehicles can be damaging to quality of life and human safety [16]. However, it is difficult to secure solutions based on heterogeneous robots, since although different manufacturers offer particular solutions that focus on specific security aspects (e.g., DJI's Pilot PE app [17] or Parrot's FreeFlight 6 app [18], which ensure data privacy), there is a lack of global solutions that cover all areas of security and there is no universal solution valid for heterogeneous ecosystems.

The solution presented by the authors of this paper is based on DDS, which offers a real-time communication middleware able to establish a common connectivity framework for robots that have their own onboard SDK (Software Development Kit). However, the cost of a full-feature (i.e., able to provide security mechanisms) DDS commercial license

is expensive (around 8000 EUR/year) for most farmers. Some well-known providers are RTI, eProsima or Twin Oaks. Moreover, there are two open-source DDS implementations available: (a) the ADLink DDS Community Edition [19] that supports the OMG DDSI2 standard and ensures interoperability with other DDS implementations but lacks security support and (b) Eclipse Cyclone DDS [20], a distribution compliant with the OMG standards for interoperability and security (which guarantees authentication, access control and encryption). The authors of this article tested both implementations. Tests with the Eclipse Cyclone implementation have not been satisfactory for the ecosystem deployed: at the time of writing this article, neither flexible configuration of DDS options (e.g., QoS parameters and automatic discovery) nor compatibility with embedded systems (e.g., lack of compatibility with Raspberry Pi, a common board for unmanned robots) is supported. Tests with the DDS Community Edition have been successful from the communication point of view, but as said before, the solution lacks security support mechanisms.

2.2.4. IoT Reference Architectures for Distributed Systems

The authors of this article analysed several IoT reference architectures in order to decide the best options to provide a cyber-security design for a distributed architecture.

Microsoft's Azure architecture summarises a set of generic security considerations [21] when dealing with data exchanges between IoT devices an Azure's PaaS (Platform as a Service) components: TLS 1.2 (Transport layer Security) compatibility provision; data encryption through a secure symmetric key encryption algorithm; and device authentication through a secure method, such as Digital Signature based on a secure symmetric key encryption algorithm, pre-shared keys, X.509 certificates or unique tokens per device stored in a KeyStore. Azure also recommends implementing mechanisms for device firmware updates that enable remediation of discovered security vulnerabilities.

Similarly, Amazon's AWS (Amazon Web Services) IoT security architecture [22] is focused on securing communications between IoT devices and AWS Cloud services. Its Cloud security model proposes TLS encrypted communications between any IoT device and the Cloud, and unique credentials for every IoT device that could be established through X.509 certificates, AWS credentials, Amazon Cognito identities or custom authentication tokens. Every credential must be granted specific permissions to access Cloud services.

The Intel IoT Platform [23] implements its own tools for a layered security approach. The Endpoint Device Level focuses on device authentication and data confidentiality based on EPID (Intel Enhanced Privacy ID), which provides immutable identity and end-to-end data protection for every device. The Network Level ensures data security in transit through the use of intelligent gateway solutions developed in cooperation with McAfee. The Cloud Level focuses on protecting data centres.

Google Cloud provides specific security services, such as Chronicle [24], a service that extracts signals from the IoT telemetry of a device to find threats, or the Asset Inventory [25], an inventory tool to monitor the identity of any access to the Google Cloud Platform resources of a project.

As can be seen, all of these Cloud service providers agree on the security needs of IoT architectures (data encryption, device authentication, TLS compatibility, databases security, etc.) but there is no one-fits-all solution to follow and each of them implements its own particular solution.

3. Results

3.1. Architecture Design for Swarms of Autonomous Vehicles

Below, a proposal for a communication architecture is provided, the goal of which is to guarantee the real-time requirements of a swarm of autonomous vehicles. This type of system is composed of a set of core components, as depicted in Figure 1. First, a Mission Console, where the operator can define the missions to be executed by the swarm of vehicles, e.g., surveillance of an area, collects observations from short-range sensors that lack an internet connection, ploughing a field or applying prescription maps. The Mission

Console is usually a web or mobile application, accessible via HTTP, that can be standalone or integrated into the FMIS (Farm Management Information System). It sends the missions to the vehicles through a middleware, responsible for unifying and hiding the underlying heterogeneity of the hardware layer.

The middleware can be deployed in the Cloud or at the Edge. The authors of this study propose a Cloud approach, especially for small and medium farms, to take advantage of the benefits of Cloud computing including reduction of IT costs, data and systems protection, quick deployment of services, access to automatic updates, backup of data, reliability and ubiquitous access. Moreover, deploying the middleware in a pay-per-use Cloud platform such as AWS has the advantage that this type of Cloud platform implements cyber security and threat detection services that continuously monitor and protect deployments [26].

The middleware includes two components: the SWARM Controller and the Data Repository. The SWARM Controller component is in charge of managing real-time communications with the autonomous vehicles both to send the missions and to gather the feedback or status from the vehicles. The authors of this study recommend using a distributed publish/subscribe interface based on the OMG standard DDS in the communications among vehicles and with the SWARM Controller. The reasons for this decision are described in Section 2.2.2. On the other hand, the SWARM Controller provides a REST interface for communications with the Mission Console.

The Data Repository stores all of the data exchanged by the system. It usually consists of relational (e.g., MySQL [27]) and NoSQL databases (e.g., MongoDB [28]) that can be hosted in the Cloud, and the data can be accessed through REST APIs.

In REST services, HTTP requests made to the URI (Universal Resource Identifier) of a resource generate a response with a payload that can be formatted as plain text, in JSON (JavaScript Object Notation) or in XML (Extensible Markup Language) format. JSON is recommended for lightweight, faster communications and to reduce costs when transferring bytes over the network.

Finally, we have the farming autonomous or unmanned vehicles where we can consider tractors with their equipment and implements; harvesters that help to harvest crops efficiently; ATVs (All-terrain Vehicles) that are smaller than tractors and can move across rough terrain; UGVs, which are mobile robots equipped with technologies for positioning, navigation, planning, sensing and actuating; and aerial vehicles such as drones or UAVs.

3.2. Security Architecture for Swarms of Autonomous Vehicles

When approaching a cybersecurity design, the first step is to analyse the possible threats in all communication channels of the architecture under examination. In the case of this paper, the authors identified all of the channels susceptible to attacks in the system architecture presented in Figure 1. These channels are signalised with letters from A to E and are summarised in Table 1.

Table 1. Identification of communication channels in the architecture described in Figure 1.

Communication Channel	Architecture Components Involved
A	Operator–Mission Console
B	Mission Console–SWARM Controller
C	SWARM Controller–UAV/ATV/UGV
D	Mission Console–Data Repository
E	SWARM Controller–Data Repository

In addition to taking into account the security of the communication channels used, it is also necessary to ensure the security of the data stored in the repositories. In the case of the architecture under study, the Data Repository could consist of SQL and/or NoSQL databases.

The purpose of this article is to propose an appropriate cybersecurity design for the architecture under study, considering the security recommendations included in Section 2.2.1.

This design tackles three aspects: (a) ensuring the security of data transmission through the communication channels identified in Table 1; (b) guaranteeing the security of the data repositories; and (c) ensuring the security of processing, so that the system can be restored in the event of an incident or attack. The cybersecurity design proposal of the authors is described below.

3.2.1. Protection of Data Confidentiality

Data confidentiality is about protecting data against unintentional or unauthorised access, disclosure or theft. The main method to protect data confidentiality is data encryption. Encryption is a process that renders data unreadable to anyone except those who have the proper key or password. The purpose of data encryption is to protect digital data confidentiality as it is transmitted through computer networks or stored on computer systems. Data encryption and decryption can be performed in two ways: (a) secret-key encryption (also known as symmetric encryption), in which the sender and receiver of the communication agree on a particular secret key used to encrypt and decrypt exchanged messages between them. The security of this method relies not only on the strength of the algorithm but also on the strength of the key itself. The most popular and secure symmetric encryption algorithm to date is AES (Advanced Encryption Standard). (b) For public-key encryption (also known as asymmetric encryption), the sender and receiver have two keys each, one for encryption (public) and one for decryption (private). All senders know the public key, but the private key is owned by the receivers only. Standard asymmetric encryption algorithms include RSA (Rivest- Shamir-Adleman), Diffie-Hellman and ECC (Elliptic Curve Cryptography). Although asymmetric encryption is more effective than the previous one, it penalises communications performance as the process takes longer and may not be suitable for cases with real-time constraints. Symmetric cryptography is designed precisely for the efficient processing of large volumes of data.

The solutions and recommendations proposed to protect data confidentiality in the system under study are based on these alternatives and are described below:

- Communication channel A: with the Mission Console being a web application accessible via browser, the best encryption solution in this case would be to use TLS [29], an updated version of its now-deprecated predecessor SSL (Secure Sockets Layer). TLS is a standardised technology that allows data traffic to be encrypted between a web browser and a web server by using the HTTPS protocol (HTTP Secure). OpenSSL [30] is proposed as a solution to provide TLS, and confidentiality could be implemented using the AES symmetric encryption algorithm. Security is granted in HTTPS by using digital certificates. TLS uses X.509 digital certificates to fulfil two functions: data encryption and authentication. Once installed, an X.509 certificate verifies the authenticity of a host or site through public key cryptography. After authentication (explained in Section 3.2.2), a one-time secret key is exchanged between entities, which is used to encrypt the data flow between the parties involved throughout the session. X.509 certificates are most reliable when issued by a trusted Certificate Authority (CA), which are organisms (e.g., Verisign) that must follow very strict policies regarding who to grant a TLS certificate. CA-issued digital certificates usually have a cost. In order to ensure compatibility, the CA must be listed in the Trusted Root Certificates of the browser.
- Communication channel B: the Mission Console consumes the REST API offered by the SWARM Controller. As in the previous case, the best option to encrypt the HTTP channel between both entities would be to use TLS. An X.509 certificate must be installed on the SWARM Controller. The Mission Console must be configured to trust the CA of the certificate. Moreover, in order not to leak confidential information, a good practice when designing the REST API is to avoid transferring parameters in the URL of the services, always using HTTP request headers or bodies (e.g., HTTP POST and PUT methods) [31].

- Communication channel C: as the SWARM Controller and the deployed vehicles do not share a physical connection (i.e., the former is deployed in the Cloud, while the latter use a 3G/4G connection), using a VPN (Virtual Private Network). A VPN provides a method to communicate nodes that are not physically connected to a shared network. However, data to be transmitted over a VPN is encrypted before being sent to the destiny. The IP addresses of the VPN endpoints are hidden as well. In order not to penalise communication performance, which is something especially important in the management of autonomous vehicles, it is recommendable to select VPN solutions based on secret-key encryption, usually considered as fast and low battery consumption algorithms. LogMeIn Hamachi VPN [32] is proposed as a solution to establish the VPN, a free solution for up to five nodes deployed in the same VPN network. This VPN solution is independent from the network operator, so any SIM card can be used on the vehicle side. In this case, a key exchange protocol based on the Diffie–Hellman algorithm would take place between any two VPN nodes willing to exchange messages. Once a session key has been established, AES-256 cipher would be used for data encryption.
- Communication channel D: the data in the Repository can be accessed through REST APIs. As in the case of communication channel B, the best option for channel encryption would be to use TLS. A X.509 certificate must be installed on the server side (the Data Repository side in this case). The Mission Console must be configured to trust the CA of the certificate.
- Communication channel E: the authors of this study consider that the encryption of this communication channel is not essential, since the components involved can be deployed in a monitored Cloud that does not offer external interfaces. By using an unencrypted interface between the SWARM Controller—responsible for supervising the operations of autonomous vehicles—and the Data Repository, the real-time requirement for managing autonomous vehicles is not compromised.
- Data Repository: it is possible to encrypt data stored in a database at a number of levels: storage-level, database-level and application-level. For SQL databases, one of the most commonly used methods is column-level encryption (performed at the database level), significantly more flexible compared to entire database encryptions. Moreover, this method allows for using separate secret keys for each column within a database. For NoSQL databases, this approach may not be the most suitable, considering that not all documents define columns before inserting data. However, in general, it should be noted that encryption can cause performance degradation in all databases accessed, so it is advisable to encrypt only sensitive data (i.e., private data such as passwords, which should be stored in the database data dictionary in encrypted format).

3.2.2. User or System Authentication

The purpose of authentication is to confirm or verify the identity of a user or process. Authentication during the setup of a connection ensures that data are only exchanged between correct parties. Basic user authentication is typically based on two elements: an identity and a password. This method can prevent user logging into other people's accounts, but it is not useful in brute force attacks or password theft situations. Advanced authentication mechanisms are needed to ensure that users are who they say they are. Some of the most popular advanced authentication technologies are (a) limitation of login attempts, so it should not be possible to force an account through its web user interface (user accounts that perform many incorrect logins can be blocked, and account holders can be notified), which is simple to implement but is not effective in the case of password theft; (b) two-factor authentication, a method that requires users to enter a code sent to one of their devices before they can log in, which is common and low-cost to implement but can circumvent the security offered due to a lost device; (c) context-based authentication, which relies on information about the device and user location (any log attempt from a new location or device is notified to the account holder); (d) biometric authentication, which

relies on the unique biological characteristics of an individual to verify his/her identity (e.g., fingerprints, voice and face), which are difficult to fake but requires specialised scanning equipment; and (e) third-party authentication, in which the identity of an entity is verified by a third-party. Some examples are Certificate Authorities in the case of the X.509 digital certificate associated with digital signature, a cryptographic method that proves that data were signed by none other than the private key holder; OAuth [33] Authorisation Servers in the case of token-based authentication solutions; Google Authenticator [34] for the case of Google applications that implement a two-factor authentication mechanism; or a LDAP (Lightweight Directory Access Protocol) server. LDAP is a standard protocol that implements remote authentication through distributed directory information services. LDAP provides a central place to store usernames and passwords, which allows services to connect to the LDAP server (also known as Directory System Agent) to authenticate users and to check their role. LDAP is often the protocol of choice for many open source technical solutions, such as Docker, Kubernetes or Jenkins.

Based on these alternatives, several authentication mechanisms are recommended in the system under study:

- Communication channel A: due to its robustness and flexibility, the authors of this article suggest providing mutual authentication based on third-party mechanisms, for both users and the Mission Console. On the one hand, user authentication can be performed using LDAP, an especially suitable solution if you are looking for a centralised management of user information. OpenLDAP [35] is proposed as an open source solution to implement an LDAP server. In this case, the Mission Console acts as an LDAP client. If TLS is used to protect communication between LDAP clients and remote LDAP servers (LDAP over TLS), the exchange of user credentials remains encrypted. The process of user authentication begins with the user providing his/her credentials to the Mission Console, which validates username, password and role in the LDAP Server. If validation is successful, the user is granted access to the Mission Console, according to the assigned role. On the other hand, authentication of the Mission Console can be performed thanks to TLS and the X.509 certificate installed in the host. In the case under study, authentication can be implemented by the RSA algorithm provided by OpenSSL. When an operator (browser) sends an HTTPS request at connection setup, a TLS handshake takes place. The Mission Console responds with its certificate (containing its public key). The browser checks within its database of trusted certificate authorities, and if the authority that issued the received certificate is within that database, the browser trusts the Mission Console. Then, the browser responds with handshake data encrypted with the public key, which is decrypted by the server using its private key. Once the authentication process finishes, the browser and server negotiate a one-time encryption key to be used during the session. One-time key confidentiality is protected by TLS.
- Communication channel B: mutual authentication of the Mission Console and the SWARM Controller can be ensured by the X.509 certificates installed at both ends. To prove their identities, both the Mission Console (the REST client) and the SWARM Controller (the REST server) present their identities in the form of certificates. The client and server initiate a TLS handshake before the actual REST API messages are exchanged. In the case under study, the implementation can be based on the RSA algorithm provided by OpenSSL. Another option for basic REST Server authentication could be to use HTTP Basic Authentication, a method in which credentials (username and password) encoded in Base64 are sent directly in HTTP headers without encryption. This is a very simple authentication method, but since confidential data are transmitted in plain text, it should always be used in combination with HTTPS.
- Communication channel C: VPN endpoints must be authenticated before a VPN tunnel is established. Two types of methods are commonly implemented by VPN, and these are either using pre-shared keys or digital certificates. Pre-shared keys are usually the default option, with VPN administrators being responsible for generating

the key at configuration time. If the chosen option is certificate authentication, a possibility is to implement digital signature between the nodes. In the case under study, in order to guarantee the real-time settings needed by autonomous vehicles, it is recommended that the VPN nodes (e.g., the Hamachi VPN nodes) authenticate using pre-shared keys.

- Communication channel D: access of users of the Mission Console to the Data Repository must be validated in order to ensure identity authentication and to prevent unauthorised databases usage. As in the case of communication channel B, mutual authentication of the Mission Console and the Data Repository can be ensured by X.509 certificates installed at both ends. Moreover, database user credentials can coincide with the username and password defined for operators in the Mission Console. This is useful to guarantee non-repudiation, as will be explained in Section 3.2.5.
- Communication channel E: the authors of this study consider that middleware components can trust each other, as they can be deployed in a monitored Cloud. Thus, the SWARM Controller application can be granted access to the Data repository. Nevertheless, the credentials granted to the SWARM Controller should be generated by the Database Manager of the Cloud services provider and would only be valid if used inside the particular Cloud domain. The lack of extra security mechanisms ensures good responsiveness of access to the Data Repository, something essential from the point of view of management of autonomous vehicles that require real-time communication.

3.2.3. Protection of Data Integrity

In cybersecurity, data integrity refers to the accuracy of the data. The objective of the techniques that guarantee the integrity of the data is to prevent the data from being modified or misused by an unauthorised party. Altered data in agriculture could cause damages to livestock (e.g., withholding feed or water) or crops (e.g., applying fertilizer in an area of risk) [7]. Data validation is another requisite for data integrity, so it is important to check the accuracy, completeness and consistency of data: e.g., discarding incomplete or out-of-range data. The most common tool for ensuring data integrity is HMAC (keyed Hashed Message Authentication Code), which uses symmetric cryptography to reduce data to a message digest (i.e., a short string of numbers). A secret key is randomly generated by the inherent HMAC algorithm, and this key must only be shared with trusted parties. The hash is unique to the information being processed, so any change in the data produces a completely different message digest. The message digest must be sent together with the data. By comparing the digest received with the digest generated from the incoming data, the recipient of the data ensures that the information remains unaltered. The strength of the HMAC depends on the cryptographic strength of the hash function and on the quality and size of the secret key. TLS provides an implementation of HMAC based on the cryptographic functions MD5 and SHA-1. Digital signature is another option for data integrity. The difference between HMAC and digital signature is that HMAC values are generated and verified using the same secret key, which implies that the sender and receiver agree on the same key before initiating communications, which provides a faster performance than digital signature.

Another option to ensure data integrity is the use of Blockchain, a technology based on a decentralised-shared database that records transactions in an immutable ledger. Each of these transactions is a block of the blockchain and every block contains information about the transaction time, its details and previous transactions. Once recorded, a block cannot be altered, which ensures the integrity of the data stored in the blockchain. As a drawback of this technology, it can be mentioned that its implementation is a costly process. Moreover, there still exist some regulatory barriers that have to be removed [36].

The measures recommended to guarantee the integrity of the data in the case under study are described below:

- Communication channel A: the integrity of data exchanged between the operator's browser and the Mission Console can be ensured by TLS (and implemented by

OpenSSL). In this case, message transport should include a message integrity check based on SHA-1 (HMAC algorithm). The secret key is randomly generated by the inherent HMAC algorithm. Per each of the requests to the Mission Console, the client creates a unique HMAC by hashing data requested with the secret key and sending it together with the request. Once the Mission Console receives the request, it calculates its own unique HMAC and compares it with the one received. If they coincide, the client is trusted and the request is executed.

- Communication channel B: as in the case of communication channel A, the integrity of this data link can be guaranteed by TLS. For JSON-based message exchanges where TLS is not possible, it is recommended to use JOSE (JavaScript Object Signing and Encryption). JOSE is a framework that provides a collection of specifications to ensure cybersecurity in JavaScript-based message exchanges. JOSE is able to provide not only integrity but also confidentiality, authentication and non-repudiation. JOSE guarantees integrity through HMAC algorithms based on SHA-2. In addition to data integrity checks, in the case of JSON data formats, it is also interesting to validate the data against its schema. A JSON Schema [37] is a grammar language to validate the syntax of a JSON document through a set of metadata about the meaning and valid values of the object's properties. Regarding data validation, JSON schemas should be defined to establish the structure and content of all JSON objects to be exchanged.
- Communication channel C: VPNs implement hashing mechanisms to ensure that data have not been intercepted and altered when travelling from one VPN endpoint to another. In the case under study, the proposed Hamachi VPN checks the integrity of all data exchanged by using HMAC-SHA-1.
- Communication channel D: as in the case of the communication channel A, the integrity of this data link can be guaranteed by TLS.
- Communication channel E: as expressed in the previous sections, the authors of this study consider that the components of the middleware can trust one another as they can be part of the same monitored Cloud.
- Data Repository: a good practice to ensure that information can be restored in case its integrity is compromised is the implementation of regular data backups. Moreover, databases could include data integrity checks based on fixed schemas or predefined sets of rules. For example, columns in a relational database could be forced to be declared upon a defined domain (e.g., semantically established in an ontology or taxonomy) to abstract the heterogeneity and to ensure that all information is stored according to a common information model that guarantees interoperability or is validated by regular expressions specifying the search patterns. In the case under study, it is recommended that all data entered by operators through the Mission Console is checked by regular expressions.

3.2.4. Protection against Unauthorised Users

Access control or authorisation is a cybersecurity method that regulates which users (i.e., human beings, autonomous agents, etc.) can have access to resources in a computing environment. Although closely related to authentication, the purpose of authorisation is different. While authentication confirms the identity of users, authorisation assigns roles and permissions to users once they have been authenticated. Authorisation on its own is not sufficient, and it is also important to determine what actions, if any, authenticated users are authorised to perform. One of the most common access control models today is Role-Based Access Control (RBAC), which ensures that only authorised users are given access to certain resources. A role is associated with specific responsibilities, while a permission is an authorisation to carry out an action. A role can have multiple permissions associated with it. System administrators can create roles, can grant permissions to those roles and can assign users to those roles. A violation of role assignation could be detected by machine learning techniques, e.g., advance classifiers based on competitive learning [38].

In the case under study, various mechanisms are proposed to be implemented to control user access and role assignation:

- Communication channel A: in this case, the LDAP server proposed would be responsible for granting access to the operator after the user has been authenticated. As explained in Section 3.2.2, LDAP servers include not only user identification information but also assigned groups and network access rights. The process starts when the user provides his/her credentials to the Mission Console for validation. Once validated, the user is granted access to the Mission Console according to the defined role.
- Communication channel B: the management of user access to the REST API offered by the SWARM Controller can be addressed from different points of view: (a) restricting supported HTTP methods, so that connecting clients have access to the minimum capabilities required for the service only; (b) using HTTP security headers to specify the security-related details of HTTP communication, which provides a good method to protect a web application against common attacks such as Cross-Site Scripting (XSS) [39] and Clickjacking [40]; (c) validating permitted content types, specifying allowed types in both the Content-Type and the Accept header (including the charset is also a good practice); (d) validating input data so that any request that does not conform to the API specification is rejected; and (e) defining API keys that must be provided by clients in order to consume the API. This is common for monetised services.
- Communication channel C: user authorisation is managed by a VPN. VPN authorisation is responsible for verifying that users have the authority to access a specified network, granting the requested access when appropriate. In the case under study, users (i.e., the SWARM Controller and all vehicles) must register in the VPN provider website to be granted credentials that guarantee access to the specific Hamachi private network to be deployed.
- Communication channel D: authorisation of access to the Data Repository is a privilege provided by the database manager, who is responsible for defining which database operations a user can perform. Privileges can be granted to individual users, to groups or to all users. For example, in SQL databases, users can be granted read, insert, update or delete privileges. Privileges are stored in the database catalogues. As explained in the introduction to this section, one of the most common methods for authorisation management is RBAC, which consists of assigning each database record with a set of permissions linked to roles. The privileges of the users that are members of a role are defined by the role. In the case of the users of the Mission Console, they are granted database privileges either to view or to add new content to the Data Repository depending on their role.
- Communication channel E: as in the case of the communication channel D, the SWARM Controller must be granted both privileges to view and to add new content to the Data Repository.

3.2.5. Non-Repudiation

Generally, non-repudiation implies the association of changes or actions with a unique individual. This usually implies that the sender of information receives proof of delivery and that the recipient receives both proof of sender's identity and data integrity confirmation, so that no communication party can subsequently deny either the origin, authenticity or integrity of the data, or having processed the information. It is closely related to authentication, with authentication being a technical concept and non-repudiation being a legal concept. That being said, it is important to highlight that true non-repudiation is something that is not always easy to achieve because, even if a technical solution is deployed to uniquely identify the executor of any action, malware-type attacks can occur, with malware's final purpose being to steal user credentials and to perform actions on their behalf. Therefore, the first important step to achieve non-repudiation is to use a good and up-to-date antivirus software as an antimalware protection system [41].

In the case of databases, the non-repudiation guarantee is sometimes not easy to achieve either. Most database solutions have audit functionalities available, so that the database manager can audit database activities such as insert, update, delete or select. However, this is not useful for shared schemas or multi-tiered applications, in which the application user is not the same as the database user. For those cases, it could be more interesting to rely on additional fine-grained audit functionalities usually provided by the database solution able to store activity information with great detail. Nevertheless, it must be noted that this type of solution requires large amounts of memory and could be cost prohibitive.

Taking all these considerations into account, the measures suggested to guarantee non-repudiation in the case under study are as follows:

- Communication channel A: unique identification of users could be performed via LDAP, responsible both for user authentication and authorisation. The Mission Console authentication would be granted by its X.509 certificate;
- Communication channel B: mutual authentication of the Mission Console and the SWARM Controller could be guaranteed by the X.509 certificates installed at both ends. Though the REST API implementation provided by the SWARM Controller does not cover user identification, all user actions (including those related to the definition of agricultural missions for vehicles) are recorded in the Data Repository through the communication channel D;
- Communication channel C: unique identification of VPN nodes connected to the proposed Hamachi VPN server would be guaranteed by the user credentials needed to register in a particular VPN and by the pre-shared keys (defined at registration time by the VPN administrator) that nodes must provide to be trusted by the rest of their peers;
- Communication channel D: mutual authentication of the SWARM Controller and the Data Repository would be guaranteed by the X.509 certificates installed at both ends. This communication channel is used to record all user actions (e.g., agricultural missions for vehicles). Tracking of user actions would be guaranteed as the Mission Console uses the specific credentials of each user (i.e., the ones validated by the LDAP Server) to access the Data Repository. This information is duplicated in the log files recorded by the Mission Console;
- Communication channel E: the authors of this study consider that the identity of the SWARM Controller and the Data Repository is guaranteed by the monitored Cloud. This communication channel is used to record vehicle actions (i.e., vehicle status and mission status). Tracking of vehicle actions is guaranteed as all vehicle feedback is recorded in the database with the specific vehicle identification, and this information is duplicated in the log files recorded by each vehicle.

3.2.6. Security of Processing

Security of process is related to the ability of a system to restore availability in a timely manner in the event of an incident or attack. It must be ensured that the system operates reliably, under normal and abnormal production conditions, preventing denial-of-service situations.

To achieve this goal, all system software must be backed up accordingly with up-to-date data to provide a recovery from a software failure, misconfiguration or software change due to malicious malware. Data stored in repositories must be backed up, e.g., using generic tools to back up systems. It is recommended to keep the backup copies in a safe place for a period to be determined. The source code and image of the system components must be stored in a repository, e.g., GitLab, allowing system restoration in the event of a disaster.

The software update process should be monitored to perform an error-free firmware update. The authors of this article propose to deploy an intermediate buffer for data

backups prior to firmware updates. It must be ensured that only after a correct software update can the new software version become active.

4. Discussion

The work presented in this paper is based on the hypotheses that a swarms of heterogeneous aerial and terrestrial vehicles can collaborate together in agriculture scenarios by exchanging information and by combining their capabilities, advanced sensors and equipment to carry out complex missions in a more time- and cost-efficient manner.

Such a system needs to manage secure communications to elude cyber security attacks or malicious use of vehicles; multi-node real-time communications, reliability and QoS management for low energy consumption; and interoperability among heterogeneous vehicles and data formats for data exchange comprehension.

Therefore, the results and security architecture presented in this paper covering confidentiality, integrity, authentication, access control or authorisation and non-repudiation are of interest to any system where two or more entities or nodes communicate and transmit information across a network.

If, in addition, these entities are large in number or part of a critical system, which means that the system must be highly reliable and must process the data in real-time with well-defined and fixed time constraints, then a distributed DDS-based communication system is recommended over socket-based solutions or other publish/subscribe mechanisms based on central brokers.

Finally, there are several distributions of the DDS standard. However, commercial ones are too expensive for farmers and small companies. On the other hand, open-source licenses do not offer embedded security mechanisms or dynamic discovery over different networks. Hence, the security implementations and recommendations described in this paper make up for this lack, whereas the VPN approach adds an additional security layer and provides a shared network where dynamic discovery is possible with open-source solutions.

5. Conclusions

The authors of this article propose an end-to-end cybersecurity design addressing systems that manage missions with swarms of heterogeneous autonomous or unmanned vehicles in Smart Farming. This design covers all of the aspects recommended by the specification ISO 7798-2.

Figure 2 presents the main cyber-security solutions proposed for the architecture under consideration, i.e., an LDAP Server, a VPN, TLS and X.509 certificates.

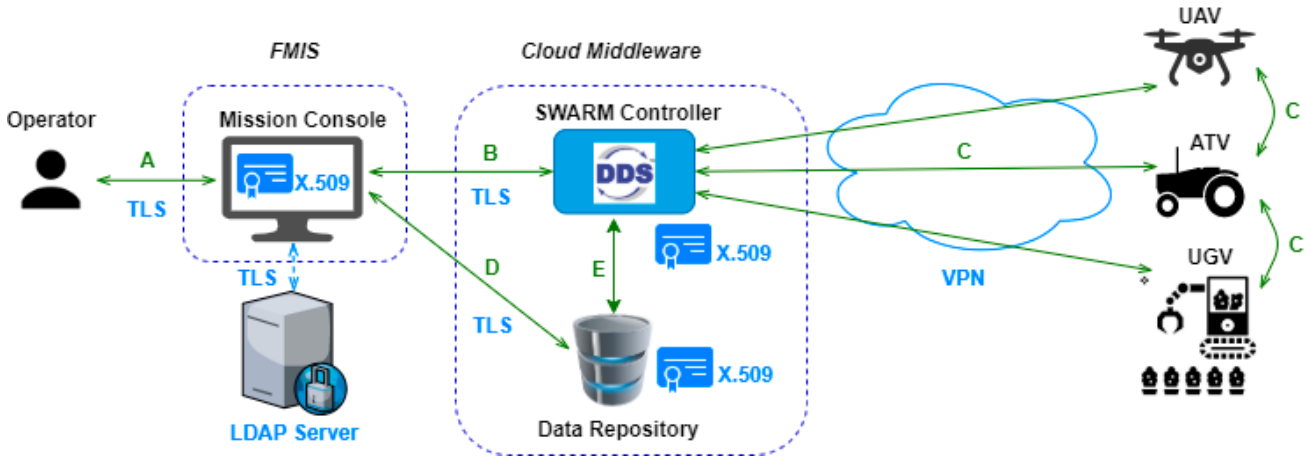


Figure 2. Security solution for a system to manage missions with swarms of autonomous vehicles.

The use of an LDAP Server is a good solution for authentication, authorisation and non-repudiation of users, as it is widely supported across all industries (i.e., multi-platform

and multi-vendor-compatible implementations can be found), and it is extensible to meet future requirements. In this case, the Mission Console would check all user credentials and roles in the LDAP Server.

The combination of TLS and X.509 certificates guarantee confidentiality, authentication, integrity and non-repudiation over HTTP networks. TLS implements a combination of symmetric and asymmetric cryptography so it is possible to find the right balance between performance and security of communications. X.509 certificates must be provided by recognised Certificate Authorities.

The deployment of a VPN to communicate with vehicles provides an integral solution to secure the connection through a WAN (Wide Area Network). A large number of VPN implementations can be found nowadays but note that the security suite offered by each of them may vary. When choosing a VPN product, it is important to carefully analyse the security features that the particular solution offers.

A more detailed summary of all of the cybersecurity measures recommended in this paper is presented in Table 2.

Table 2. Summary of the cybersecurity solutions proposed for the target architecture.

Cybersecurity Goal	Entities Involved	Cybersecurity Technology
Data confidentiality: encryption through symmetric cryptography.	Communication channel A Communication channel B Communication channel C Communication channel D Communication channel E Data Repository	HTTPS over TLS (AES-CBC). HTTPS over TLS (AES-CBC), HTTP POST or PUT methods for SWARM Controller REST API. VPN guaranteed: AES-256. HTTPS over TLS (AES-CBC). N/A. Encryption of sensitive data only (e.g., passwords).
Authentication: LDAP, asymmetric cryptography, credentials for VPN users.	Communication channel A Communication channel B Communication channel C Communication channel D Communication channel E	LDAP (users), X.509 certificate + RSA algorithm (Mission Console). X.509 certificate + RSA algorithm (Mission Console and SWARM Controller). VPN guaranteed: pre-shared keys. X.509 certificate + RSA algorithm (Mission Console and Data Repository). N/A.
Data integrity HMAC, database backups, data patterns and information model.	Communication channel A Communication channel B Communication channel C Communication channel D Communication channel E Data Repository	TLS (SHA-1). TLS (SHA-1). Definition of JSON Schemas. VPN guaranteed: HMAC-SHA-1. TLS (SHA-1). N/A. Regular data backups, regular expressions to check data patterns, common information model.
Authorisation: LDAP, VPN user credentials, DB roles, REST API access restrictions.	Communication channel A Communication channel B Communication channel C Communication channel D Communication channel E	LDAP (users). HTTP security headers, validation of content-type and input data, minimise allowed HTTP methods for the SWARM Controller REST API and API keys definition. VPN guaranteed: VPN user credentials. DB privileges for the users of the Mission Console based on their role. DB privileges for the SWARM Controller.

Table 2. *Cont.*

Cybersecurity Goal	Entities Involved	Cybersecurity Technology
Non-repudiation: antivirus, LDAP, X.509, user credentials, tracking of user/vehicle actions.	Communication channel A Communication channel B Communication channel C Communication channel D Communication channel E	Antivirus, LDAP (user), X.509 certificate (Mission Console). Antivirus, X.509 certificates (Mission Console, SWARM Controller). Antivirus, VPN user credentials and pre-shared keys. Antivirus, X.509 certificates (Mission Console, Data Repository), duplicated tracking of user actions (DB and log files), DB user credentials coincident with operator credentials. Antivirus, duplicated tracking of vehicle actions (DB and log files).
Security of process	All	Data repositories backup, source code and image of system components backup copy.

As future work, a study on new editions of Eclipse Cyclone DDS is contemplated. The purpose will be to check the security implementation offered, the support to embedded systems, the possibility of flexible DDS configuration and its performance in critical systems. New, open source distributions of DDS issued by ADLink will be also analysed to verify possible future compliance with security features. In addition, the authors will test different VPN solutions to compare their performance, vehicle battery consumption and potential impacts on communication latency for DDS-based architectures with real-time constraints.

Author Contributions: Conceptualisation, B.M.-R. and S.B.-A.; validation, F.J.-H. and B.M.-R.; investigation, B.M.-R., S.B.-A. and F.J.-H.; writing—original draft preparation, B.M.-R. and S.B.-A.; writing—review and editing, B.M.-R., S.B.-A. and F.J.-H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the ECSEL JU (H2020–EU.2.1.1.7.–ECSEL RIA) and the Spanish Ministry of Economic Affairs and Digital Transformation, grant number 783221—AFarCloud: Aggregate Farming in the Cloud.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. United Nations, Department of Economic and Social Affairs, Population Division. Highlights. In World Population Prospects 2019. Available online: https://population.un.org/wpp2019/ (accessed on 17 June 2020).
2. United Nations, Department of Economic and Social Affairs, Population Division. Ten Key Findings. In World Population Prospects 2019. Available online: https://population.un.org/wpp2019/ (accessed on 17 June 2020).
3. Ritchie, H.; Roser, M. Urbanization. Published online at OurWorldInData.org. 2018. Available online: https://ourworldindata.org/urbanization (accessed on 11 March 2021).
4. Hands Free Hectare. Available online: https://www.handsfreehectare.com/ (accessed on 11 March 2021).
5. John Deere's Self-Driving Tractors. Available online: https://www.gpsworld.com/use-of-autonomous-vehicles-in-mining-and-farming-touted-at-ces-2021/ (accessed on 11 March 2021).
6. Yaacoub, J.P.; Noura, H.; Salman, O.; Chehab, A. Security analysis of drones systems: Attacks, limitations, and recommenda-tions. *Internet Things* **2020**, *11*. [CrossRef]
7. Window, M. Security in Precision Agriculture: Vulnerabilities and Risks of Agricultural Systems. Master's Thesis, Luleå Uni-versity of Technology, Luleå, Sweden, 2019.
8. Sleem, L.; Noura, H.N.; Couturier, R. Towards a secure ITS: Overview, challenges and solutions. *JISA* **2020**, *55*. [CrossRef]
9. Boukoberine, M.N.; Zhou, Z.; Benbouzid, M. A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects. *Appl. Energy* **2019**, *255*. [CrossRef]

10. AFarCloud Demonstrators. Available online: http://www.afarcloud.eu/demonstrators/ (accessed on 10 May 2021).
11. Inside DDS Webinar Series. Available online: https://www.rti.com/news/rti-inside-dds-webinar-series (accessed on 18 March 2021).
12. DDS in Military Vehicles. Available online: https://iotcareer.adlinktech.com/_resx/storage/104cee9a-ie734/dds%20in%20military%20vehicles.pdf (accessed on 13 April 2021).
13. European Space Agency—Human Robot Interaction Laboratory. Available online: https://www.rti.com/hubfs/_Collateral/Customer_Snapshots/rti-customer-snapshot-esa.pdf (accessed on 14 April 2021).
14. NASA Human Exploration Telerobotics. Available online: https://www.omg.org/hot-topics/documents/dds/NASA_Telerobotics.pdf (accessed on 13 April 2021).
15. NASA Leverages RTI Middleware for Next-Generation Lunar Robots. Available online: https://www.rti.com/hubfs/docs/NASA_Human_Robotic_Systems.pdf (accessed on 14 April 2021).
16. Kim, K.; Kim, J.S.; Jeong, S.; Park, J.H.; Kim, H.K. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Comput. Secur.* **2021**, *103*. [CrossRef]
17. DJI Pilot PE. Available online: https://www.dji.com/downloads/djiapp/dji-pilot-pe (accessed on 14 April 2021).
18. Benefits of Parrot FreeFlight 6 App. Available online: https://www.parrot.com/en/newsroom/ff6-audit (accessed on 14 April 2021).
19. The ADLink DDS Community Edition. Available online: https://www.adlinktech.com/en/data-distribution-service-dds-community (accessed on 15 April 2021).
20. Eclipse Cyclone DDS. Available online: https://projects.eclipse.org/projects/iot.cyclonedds (accessed on 15 April 2021).
21. Azure IoT Reference Architecture. Available online: https://docs.microsoft.com/azure/architecture/reference-architectures/iot (accessed on 16 April 2021).
22. AWS IoT Security. Available online: https://docs.aws.amazon.com/es_es/iot/latest/developerguide/security.html (accessed on 16 April 2021).
23. The Intel IoT Platform. Available online: http://d885pvmm0z6oe.cloudfront.net/hubs/intel_80616/assets/downloads/general/Architecture_Specification_Of_An_IOT_Platform.pdf (accessed on 16 April 2021).
24. Google Chronicle. Available online: https://chronicle.security/products/platform/ (accessed on 16 April 2021).
25. Google Asset Inventory. Available online: https://cloud.google.com/asset-inventory (accessed on 16 April 2021).
26. Tawalbeh, L.; Muheidat, F.; Tawalbeh, M.; Quwaider, M. IoT Privacy and Security: Challenges and Solutions. *Appl. Sci.* **2020**, *10*, 4102. [CrossRef]
27. The MySQL Database. Available online: https://www.mysql.com/ (accessed on 11 March 2021).
28. The MongoDB Database. Available online: https://www.mongodb.com/ (accessed on 11 March 2021).
29. Rescorla, E. *RFC 8446—The Transport Layer Security (TLS) Protocol Version 1.3*; Internet Engineering Task Force: Fremont, CA, USA, 2018.
30. OpenSSL, Cryptography and SSL/TLS Toolkit. Available online: https://www.openssl.org/ (accessed on 10 March 2021).
31. How to Ensure REST API Security—Web Security Blog. Available online: https://www.netsparker.com/blog/web-security/rest-api-web-service-security/ (accessed on 19 April 2021).
32. LogMeIn Hamachi VPN. Available online: https://www.vpn.net/ (accessed on 11 March 2021).
33. Denniss, W.; Bradley, J. *RFC 8252—OAuth 2.0 for Native Apps*; Internet Engineering Task Force: Fremont, CA, USA, 2017.
34. Get Verification Codes with Google Authenticator. Available online: https://support.google.com/accounts/answer/1066447 (accessed on 25 February 2021).
35. OpenLDAP, Open Source Implementation of the Lightweight Directory Access Protocol. Available online: https://www.openldap.org/ (accessed on 10 March 2021).
36. Priem, R. Distributed ledger technology for securities clearing and settlement: Benefits, risks, and regulatory implications. *Financ. Innov.* **2020**, *6*. [CrossRef]
37. Carion, U. *JSON Schema Language Draft-Json-Schema-Languahe-00*; Internet Engineering Task Force: Fremont, CA, USA, 2019.
38. Li, T.; Kou, G.; Peng, Y.; Shi, Y. Classifying with adaptive hyper-spheres: An incremental classifier based on competitive learning. *IEEE Trans. Systems Man Cybern. Syst.* **2020**, *50*, 1218–1229. [CrossRef]
39. Cross-Site Scripting. Available online: https://owasp.org/www-community/attacks/xss/ (accessed on 5 March 2021).
40. Clickjacking. Available online: http://www.sectheory.com/clickjacking.htm (accessed on 8 March 2021).
41. 2020 State of Malware Report. Available online: https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report-1.pdf (accessed on 18 March 2021).