TYPE Hypothesis and Theory
PUBLISHED 10 May 2023
DOI 10.3389/frcmn.2023.1122231

Employing automotive security to improve the security of unmanned aerial vehicles

Kyusuk Han*

Technology Innovation Institute, Abu Dhabi, United Arab Emirates

Check for updates

OPEN ACCESS

EDITED BY
Antonio Di Maio,
University of Bern, Switzerland

REVIEWED BY
Eric Samikwa,
University of Bern, Switzerland
Hanen Idoudi,
Manouba University, Tunisia
Negar Emami,
University of Bern, Switzerland

*CORRESPONDENCE
Kyusuk Han,
kyusuk.han@tii.ae

RECEIVED 20 December 2022
ACCEPTED 03 April 2023
PUBLISHED 10 May 2023

CITATION
Han K (2023), Employing automotive security to improve the security of unmanned aerial vehicles.
Front. Comms. Net 4:1122231.
doi: 10.3389/frcmn.2023.1122231

The concern about cyber threats on unmanned aerial vehicles (UAVs) increases as they become more sophisticated and widely used in various areas. Although there have been efforts to improve the security of UAVs, their focus is limited to physical attacks and securing communication, and many areas remain open to issues. Although a more comprehensive approach is required to improve the overall areas of UAVs, it is reasonable to investigate the case study from the perspective of other similar cyber-physical systems (CPSs). Thus, we see the architectural similarity between UAVs and automobiles. Automotive security has seen significant improvement throughout the last decade as vehicles have begun to have connectivity and autonomy. There have been extensive research and development efforts in various aspects, including securing the components, securing communications, securing software updates, and securing the overall management system to provide multi-layered security for the automotive environments, and standardization and regulations are being issued. Due to the similarity, the employment of ideas from the automotive environments in the UAV environments becomes a reasonable approach. In this paper, we show the automotive security trends and discuss how UAV environments are comparable to automotive environments. We then show how security in automotives is adapted in UAVs with a case study.

KEYWORDS

automotive cybersecurity, UAV, security, vehicle cybersecurity, software update security, HPSE

1 Introduction

Unmanned aerial vehicles (UAVs) have been widely used in various areas for civil and military purposes (Ghamari et al., 2022), and they are employing more sophisticated functions beyond flight control. Although certain simple UAVs may be controlled by a hobbyist with a remote controller over a low-frequency radio channel, a sophisticated UAV may communicate with the ground control station (GCS) with more automated controls over wireless communication channels, including cellular communications for various applications, such as delivery services. Not limited to a mission using only a single UAV, even multiple UAVs could be used for various large-scale missions with the emerging wireless mesh communication among UAVs. Controlling multiple UAVs could be carried out by clustering such UAVs and letting a UAV become the cluster head to control the other UAVs in the cluster. Thus, the internal system architectures of UAVs are becoming more complex, and multiple communication mechanisms are being integrated.

However, such advances inevitably brought the concern of cyber threats, and cybersecurity has been emerging as one of the important issues in UAV environments. For example, wireless communication channels are identified as the attack surfaces threatening the security of the UAVs.

Although several studies have been conducted on securing communications among UAVs and infrastructures and physical attacks on UAVs (Wang et al., 2021; Chamola et al., 2021), many open problems remain to be considered for UAVs. As the UAV is more exposed to attackers, attackers could have more advantage in performing attacks. For example, attackers could capture a flying UAV and try to disclose confidential data or modify the system of the UAV. Protections against unauthorized physical access should be considered, such as temper protection and management of confidential information inside the UAVs. Thus, multi-layered approaches should be considered to provide security in UAV environments. With the urgent need for cybersecurity and privacy by the rapid advance in UAV applications, it is reasonable to refer to the achievements from similar cyber-physical system (CPS) domains.

Considering that a sophisticated UAV could consist of multiple components, such as a flight controller, a mission computer, and other motors and sensors interconnected, we see the similarity of architectures between automotives and UAVs. A modern vehicle consists of multiple electronic control units (ECUs), which control the powertrain, chassis, and infotainment. ECUs are interconnected via in-vehicle networks (IVNs), such as CAN, LIN, FlexRay, and Ethernet. Certain ECUs, called telematics control units (TCUs), communicate with external entities, such as other vehicles and infrastructures, and are connected to the IVNs. With the emerging trend of connective vehicles, TCUs are becoming one of the most important components in the vehicle, which enable vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communications over cellular communications or dedicated short-range communications (DSRC).

As automobiles are directly related to the safety of humans, including drivers, passengers, and pedestrians, such advances inevitably bring concerns about privacy and security. There have been various efforts to improve automotive cybersecurity in the early stage. For example, EVITA projects had been operated since 2008 and produced research work on hardware security module (HSM) specifications, V2V/V2I security design, and risk assessments when the project ended in 2011. Such work did not remain only a research topic. However, there have been extensive efforts made from every aspect, including international standards and regulations, to provide an adequate level of security and privacy.

Therefore, in this paper, we investigate the security trends in the automotive domain, which share many similarities with UAVs and have been one of the most significantly advanced over the last decade with the advanced concepts of connected and autonomous vehicles, and find the opportunities to adopt automotive cybersecurity into UAVs. We review the current state of the art of automotive cybersecurity efforts and the case study to identify the different environments between vehicles and UAVs and show how to overcome the difference. We also discuss how both domains can be integrated as more CPS applications.

The outline of this paper is organized as follows: We first compare the environments of automotives and UAVs in Section 2. We discuss the current state of the art of automotive security in Section 3 and the status of UAV security and how automotive security efforts can be employed in UAVs in Section 4. Then, we introduce the case study of employing automotive security for the UAV security use case in Section 5. We conclude the paper in Section 6.

2 Comparison of environments

As UAVs are becoming more sophisticated, the architecture of UAVs is becoming more similar to the architecture of automobiles. In this section, we first briefly show the automotive and UAV architectures in sequence and then discuss how UAV environments are comparable to automotive environments.

2.1 Automotive environments

2.1.1 Automotive E/E architecture

An automobile consists of electrical/electronic (E/E) architectures to handle complex driving operations. The E/E architecture (Figure 1) of a vehicle consists of more than a hundred ECUs to handle the operations for powertrains, chassis, and infotainment services. Certain ECUs are used only for simple work, whereas more sophisticated ECUs are being employed as advanced driver-assistance systems (ADAS) and autonomous vehicles have emerged.

2.1.2 Vehicular communication protocols

Vehicular communications consist of in-vehicle communication networks (IVNs) to interconnect ECUs and external communication networks to connect the vehicle to external entities.

There are various types of IVNs, such as the controller area network (CAN), the local interconnect network (LIN), FlexRay, or Ethernet, as depicted in Table 1. Multiple IVNs are used in the vehicle for different purposes with different capabilities.

In addition to IVNs for communicating among the internal components, various external communication methods are used for communication through TCUs and ECUs. Although several wireless communication technologies, such as Bluetooth and Wi-Fi, have already been used to connect mobile devices to the vehicle, DSRC/wireless access for vehicular environments (WAVE) and cellular-V2X (C-V2X) communications are being employed to support emerging V2V and V2I applications.

2.2 UAV environments

UAVs have various types, which could be categorized by purpose or size. For example, various sizes of UAVs (from very small to large UAVs) could be categorized by size, as depicted in Figure 2 (Burgués et al. (2019). Although these categorizations of UAVs are not directly related to their capabilities, more complexity of the system architecture could be employed in certain UAVs. We discuss how the system architectures of UAVs are designed and what kinds of communication protocols are used.

TABLE 1 Automotive in-vehicle communication protocols (Aliwa et al., 2021).

Protocol	Bit-rate	Application	Domain	Standard
High CAN	125 Kbps–1 Mbps	Critical applications	Powertrain and chassis	ISO 11898
Low CAN	5 Kbps–125 Kbps	Non-critical applications	Body domain	ISO 11898
CAN-FD	Up to 10 Mbps	Critical applications	Powertrain and chassis	ISO 11898
LIN	1 Kbps–20 Kbps	Non-critical applications	Body domain	ISO 17987
FlexRay	Up to 10 Mbps	Critical applications	Powertrain and chassis	ISO 17458

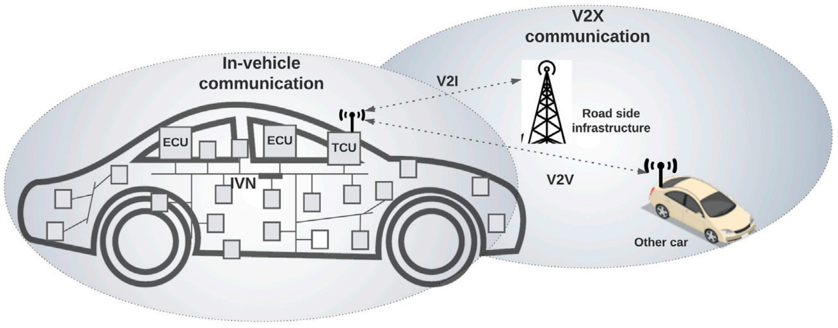

FIGURE 1
A vehicle consists of a number of ECUs, and a certain ECU has wireless connectivity, which is called a Telematics Control Unit (TCU).

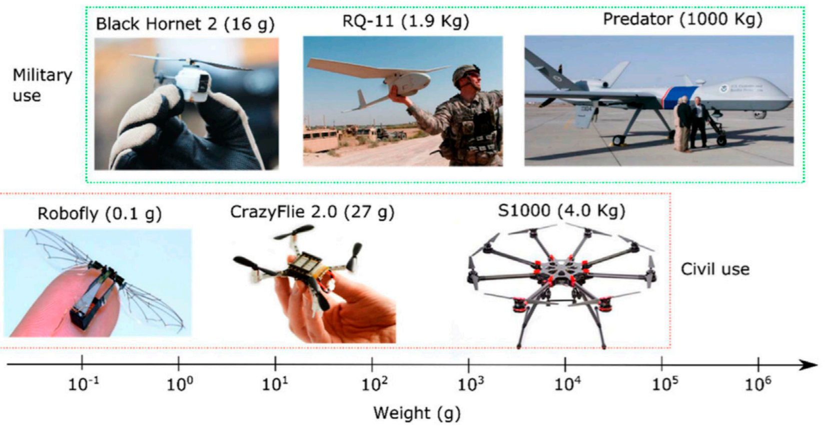

FIGURE 2
Various sizes of drones (figure from Burgués et al. (2019)).

2.2.1 UAV system architectures

Although UAV systems could consist of different components per type, every UAV consists of the flight controller as a main component, which is used to control the critical applications related to flying of the UAV, interconnecting controlling components, such as sensors and motor controllers, depicted in Figures 3A, B

As the applications of UAVs are being advanced, a dedicated mission computer is added to handle more sophisticated operations, such as object detection, collision avoidance, and prevention. Figure 3C depicts the UAV architecture with the flight controller and the mission computer.

2.2.2 UAV communication protocols

There have already been several communication methods for operating UAVs. In order to provide communication between the flight controller motors and sensors over I2C, UART, and SPI, and

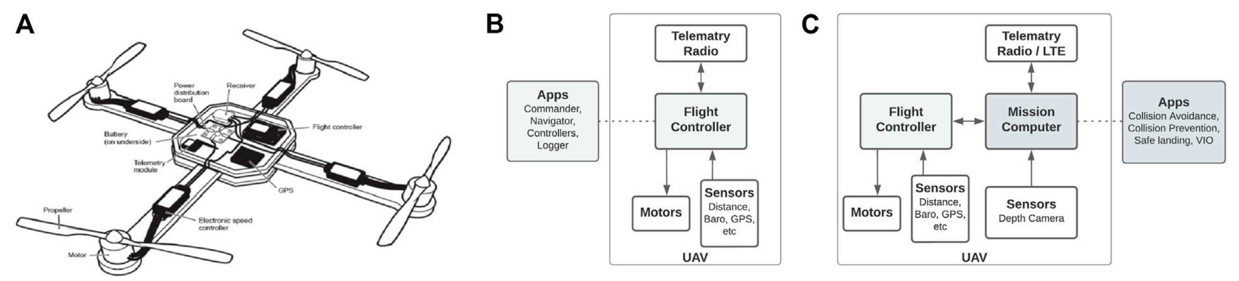

FIGURE 3
(A) Example of UAV architecture (Pothuganti et al., 2017). **(B)** UAV architecture only with the flight controller. **(C)** UAV architecture with a mission computer as a companion. **(B)** and **(C)** Simplified diagrams from PX4.

to the remote controller over radio channels, several proprietary communication formats are used for UAV communication (Khan et al., 2020), becoming similar to the automotive environment.

For example, in addition to introducing more advanced protocols, such as MAVLink (Section 2.2.2.1) and UranusLink (Kriz and Gabrlik, 2015) (Section 2.2.2.2), a CAN is used in UAV to connect more peripherals. There are protocols over CAN, such as DroneCAN (a renamed UAVCAN v1) or Cyphal (a renamed UAVCAN v2) (Section 2.2.2.3) (Lee et al., 2018). We discuss these in the following sections.

2.2.2.1 MAVLink

The Micro Air Vehicle Link (MAVLink) is one of the most widely used protocols for communication between UAVs and GCSs. It is adopted by several open-source autopilot systems, such as Ardupilot and PX4. MAVLink was developed to be a flexible, lightweight, and open-source communication protocol and used explicitly for the bidirectional data exchange between the UAV and the GCS, as well as between the UAVs. MAVLink (version 1.0) originally had 8 bytes of overhead per packet, including start sign and packet drop detection.

MAVLink did not originally implement any security mechanism; therefore, the communication channel could be vulnerable to several types of attacks, including spoofing, message forgery, and denial of services.

Although MAVLink 2.0 is extended to have 14 bytes of overhead and includes signature support to prevent the forgery of the message, it remains vulnerable to eavesdroppers. Allouch et al. (2019) clarified this vulnerability in MAVLink and proposed MAVSec, which adds encryption to MAVLink to provide confidentiality.

2.2.2.2 UranusLink

UranusLink (Kriz and Gabrlik, 2015) is a packet-oriented protocol that provides unreliable and reliable services. The protocol defined the packet structure and the data representation transmitted. Each packet consists of the following fields: preamble (PRE), sequence number (SQN), message identification (MID), data length (LEN), data field, and checksum (CS).

UranusLink introduced unsecured and secured structures. The secured structure was designed to provide confidentiality and authenticity.

2.2.2.3 DroneCAN/Cyphal

DroneCAN[1] and Cyphal[2] are open technologies for real-time intra-vehicular distributed computing and communication based on modern networking standards for manned and unmanned aircraft, spacecraft, robots, and cars. They were originally introduced as UAVCAN to provide communication over the CAN. Later, the UAVCAN v0 protocol is renamed as DroneCAN focusing on running on CAN/CAN-FD, whereas the future version of UAVCAN is renamed as Cyphal, focusing on multiple different transport-layer protocols (e.g., Ethernet and CAN).

2.3 Environment comparison

As described in Sections 2.1, 2.2, the UAV system architectures and communication models have similarities. We show the comparison in Table 2. There is a similar medium for internal communication (IC) and external communication (EC). Although the TCU connects to external entities for the vehicle, the communication module in the flight controller or the mission computer (if used) connects to external entities.

3 Security in automotive

The concept of connected and autonomous vehicles has emerged in the automotive environment since the last decade, as connectivity enables V2V and V2I communications, and autonomy enables ADAS and autonomous driving. However, those could broaden more attack surfaces exposed to various potential threats, and these threats could increase the risk that damages human safety. For example, Checkoway et al. (2011) introduced a comprehensive analysis of the possible attacks on vehicles, which are exposed to various attack surfaces through channels, as shown in Figure 4. Thus, cybersecurity has been one of the important issues in automotive domains, and there have been various efforts to provide security for the automotive.

1 https://dronecan.github.io/
2 https://opencyphal.org/

TABLE 2 Comparison between automotive and UAV architectures.

	Automotive architecture	UAV architecture
Components	Multiple (ECUs)	Single or multiple components
Number of components	+100 ECUs	1–a few
IC medium	CAN, LIN, FlexRay, Ethernet, etc.	I2C, SPI, UART, CAN, etc.
IC protocol	Proprietary on medium	DroneCAN/Cyphal (MAVLink)
EC medium	Cellular, DSRC, etc.	Cellular, Wi-Fi, etc.
EC component	TCU	Module in MC (or FC)

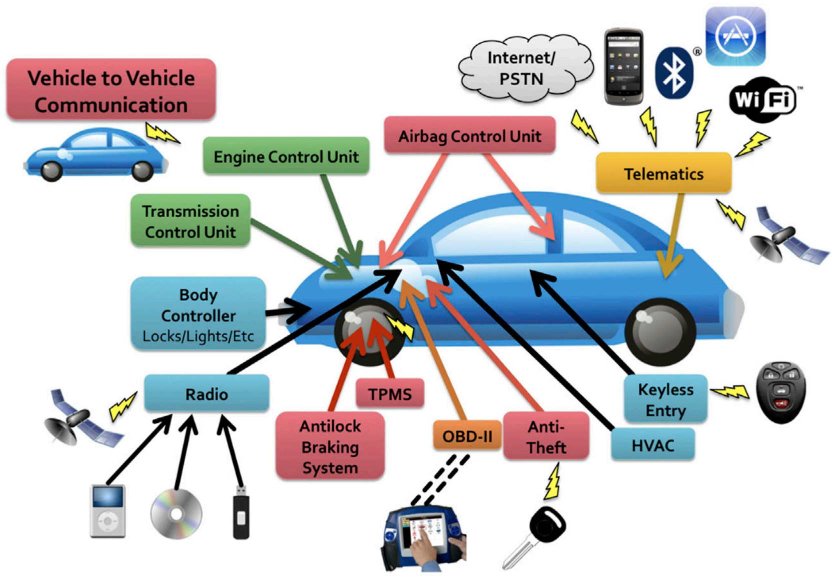

FIGURE 4
Communication channels on a modern car. Colors indicate a rough grouping of ECUs by function (figure from Checkoway et al. (2011)).

TABLE 3 Categories of security in automotive.

Category	Sub-category	Standard and regulation
Communication security	Secure V2X security	SCMS (Whyte et al., 2013)
	Secure IVN	AUTOSAR (2020)
System security	Trust anchor/HPSE	SAE International (2020); Henniger et al. (2009)
	Secure software update	Uptane (Karthik et al., 2016)
Security engineering	Secure system	SAE J3061, ISO/SAE 21434
	Software update	ISO 24089
Standards and regulations	Standardization body	SAE, ISO, AUTOSAR
	Regulatory body	UNECE (R155, R156), NHTSA

The early stage of the automotive cybersecurity research was led by the project of E-safety Vehicle Intrusion Protected Applications (EVITA) from July 2008 to December 2011. The project results include various aspects to protect the vehicle from cyber threats, including secure onboard communications, secure V2I communications, and software updates, which become the important basis to improve the cybersecurity for the automotive environments.

These results brought a strong impact on the academy and industry, and there have been various research works to improve automotive security. Table 3 shows a summary of automotive

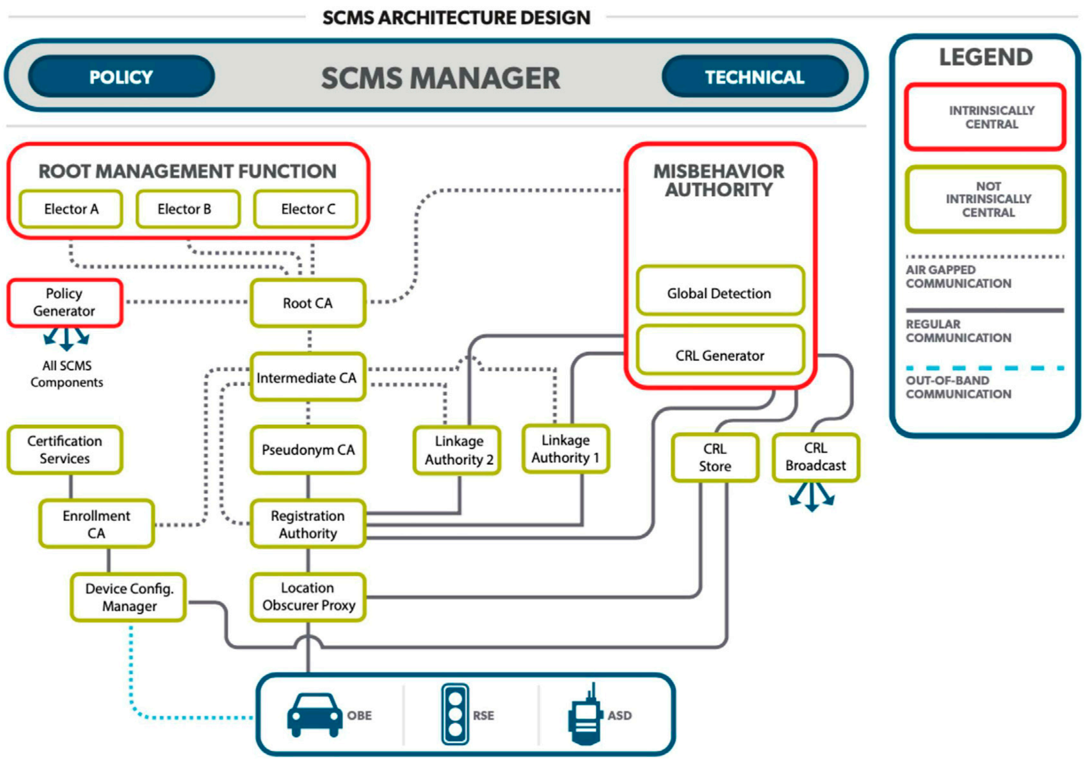

FIGURE 5
SCMS architecture design. Figure from Intelligent Transportation Systems Joint Program Office, the US Department of Transportation.

security trends in different categories, and we explain the details in the following sections.

3.1 Communication security in automotive

As a modern vehicle consists of multiple ECUs and can communicate with external entities, many researchers have focused on providing secure communication.

Various works have been proposed to secure communication with external entities, including the vehicle and infrastructures (V2V/V2I) and vehicle-to-everything (V2X) (Ghosal and Conti, 2020). Due to the importance of securing V2X communication, these works do not only remain research work; they are being adopted at the government level. The National Highway Traffic Safety Administration (NHTSA) initiated the Crash Avoidance Metrics Partnership (CAMP), which is a partnership with automotive security experts to provide a privacy-preserved security mechanism. CAMP proposed the "security credential management system" (SCMS) (Whyte et al., 2013) to enable the authentication of the vehicle safety message while preserving the privacy of the vehicle. This could be performed using anonymous but verifiable identities instead of transmitting the real identities of the vehicles, which could be tracked. NHTSA adopted SCMS for V2V/V2I environments, and Figure 5 depicts the components in the infrastructure of SCMS. For more details on components, please refer to Whyte et al. (2013).

Research on secure onboard communication among ECUs inside the vehicle has focused on providing sufficient security strength in a constrained environment. For example, a full-size HMAC-SHA256 requires 32 bytes for the message authentication code, and it may need to be truncated to be used in a CAN, which only allows up to 8 bytes of payload. Thus, many researchers (Müter and Asaj, 2011; Schweppe et al., 2011; Han et al., 2014; Nilsson et al., 2008, 2009; Matsumoto et al., 2012) proposed authentication of onboard communications, mostly on the CAN.

Each automotive vendor has also employed secure onboard communications and AUTomotive Open System ARchitecture (AUTOSAR) (AUTOSAR, 2020), which is a partnership to create and establish an open and standardized software architecture for automotive ECUs. It also provides a secure onboard communication design, as depicted in Figure 6.

In addition to protecting the authenticated frames on the bus, protecting the access by refining architectures (Kenjic and Antic, 2022), as depicted in Figure 7, is also considered. Domain or zonal controllers could operate as firewalls to prevent unauthorized access from other domains or zones.

3.2 Hardware-protected security for multi-layered security in automotive

It has been commonly agreed that protecting the security elements in the ECU only with software-based methods is

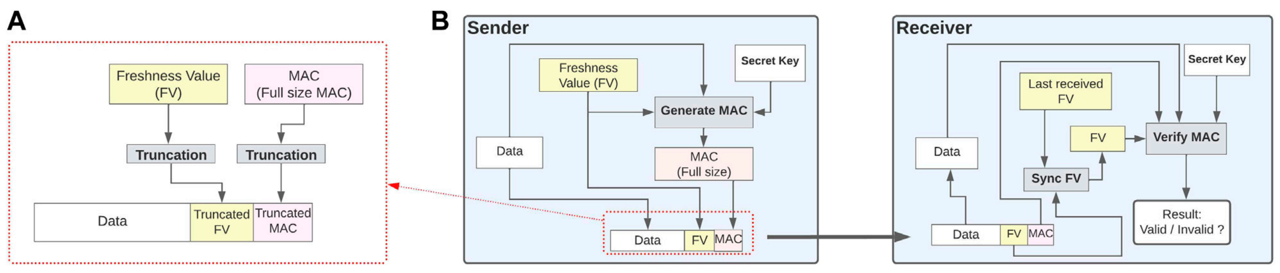

FIGURE 6
(A) Adding the truncated freshness value and authentication code in the payload. **(B)** Message authentication process between the sender and receiver. Based on data from AUTOSAR (2020).

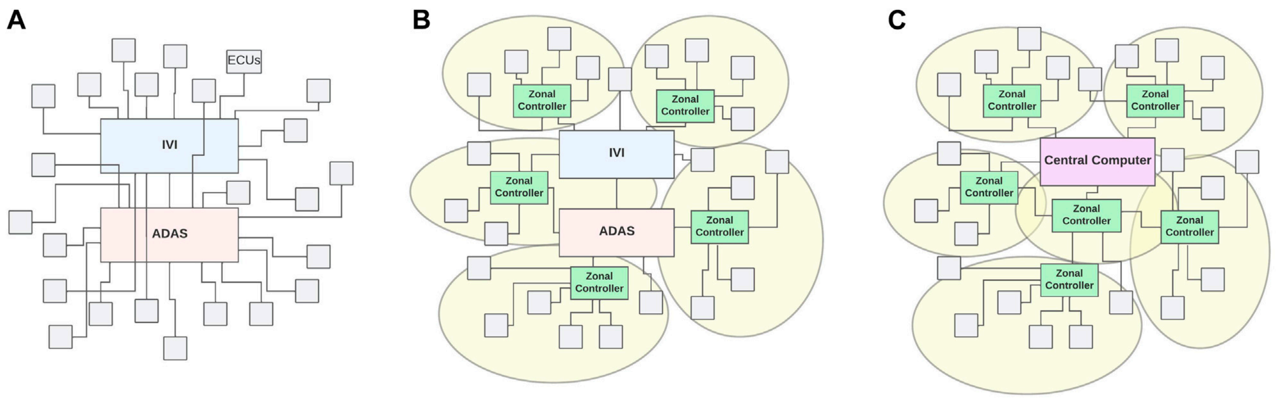

FIGURE 7
Evolved in-vehicle architecture. **(A)** Domain architecture. **(B)** Zonal architecture. **(C)** Centralized architecture. Based on data from Kenjic and Antic (2022).

insufficient, and hardware-based protection shall be deployed as the root of trust in multi-layered security (SAE International, 2020).

EVITA project defined the functional security requirements and specifications of the HSM (Henniger et al., 2009) as in the following three categories:

- **Full EVITA HSM**: Intended to provide efficient asymmetric cryptographic functions to secure V2X applications. The EVITA HSM Full Version uses its own independent internal CPU that can directly access its internal RAM and non-volatile memory to prevent any malicious interference from the application CPU and the application software, which can be accessed only by the application CPU via secure interfaces.
- **Medium EVITA HSM**: Intended to enable secure but cost-effective protection of gateways and domain controllers. EVITA HSM Medium executes very fast symmetric cryptography in hardware but rather slow asymmetric cryptography in software, such as a digital signature.
- **Light EVITA HSM**: Designed to integrate and protect ECUs, sensors, and actuators that provide or process security-critical information. This specification includes an AES hardware accelerator, with the security credentials handled by the main ECU application processor.

The Hersteller Initiative Software (HIS) consortium, founded in 2004 and consisting of members from German automotive OEMs, defined Security Hardware Extension (SHE), which is equivalent to Light EVITA HSM.

The Society of Automotive Engineers International (SAE) has been working on improving security with the hardware-assisted component. SAE defines hardware-protected security environment (HPSE)-SAE J3101 (SAE International, 2020) to avoid the confusion of HSM across different domains. Requirements of HPSE include the following categories:

1. **Cryptographic key protection:** These requirements provide protected storage, management, and usage capability for cryptographic keys.
2. **Crypto algorithm:** These requirements are intended for implementation in the HPSE to maintain confidentiality, integrity, and availability of the systems that make use of said security environment.
3. **Random number generator**: These requirements are to provide an acceptable level of random number generator.
4. **Secure non-volatile data:** These requirements consider that some cryptographic algorithms or protocols make use of retained data that need to be kept within the HPSE with non-volatile data protection (e.g., monotonic counters and bit fields).

5. **Algorithm agility:** These requirements stem from the need for a flexible design of the HPSE to accommodate updates over the lifetime of the vehicle.
6. **Interface control:** The device that implements the secure elements of the HPSE typically needs to include various hardware ports for debugging the HPSE and the ECU's functional interfaces. These hardware debugging ports could be attractive attack vectors, so their availability needs to be controlled.
7. **Secure execution environment:** The secure execution environment of the HPSE can provide separate shielding of common services, the resources used to process secret data, and various associated cryptographic logic.
8. **Self-tests:** These requirements suggest best practice recommendations for operational states and self-tests, although self-test is typically not easy to be standardized due to wide variation in microprocessor architectures and capabilities.

Using HPSE as the root of trust, various security mechanisms, including secure/authenticated boot, secure firmware update, secure communication, secure diagnosis, secure logging, and intellectual property protection, are provided to enable multi-layer protection in the system.

3.3 Secure software update in automotive

As the lifetime of a vehicle is long, there is a high possibility of new software vulnerabilities or bugs while the vehicle is in use. Thus, the software update has been one of the important features of managing the vehicle throughout its life cycle of the vehicle. Moreover, as visiting the dealer shops or repair centers to get the new software is costly, software updates over the air have been strongly required to reduce such costs. However, the ecosystem of software updates over the air could be exposed to attackers. There have been several efforts to provide secure software updates over the air in automotive environments.

The Uptane project (Karthik et al., 2016) was started in the mid-2010s to enable the secure software update over the air for the automotive. The design is inherited from The Update Framework (TUF) (Samuel et al., 2010), which is originally designed to provide security in the repository to prevent the system from installing compromised software packages. Uptane is designed to provide comprehensive security over the supply chain of software updates, considering various attacks, including server compromise, by separating roles and stages, as depicted in Figure 8. Many studies were based on Uptane (Moore et al., 2020; Qureshi et al., 2021; Al Blooshi and Han, 2022). Uptane had been standardized as IEEE/ISTO standard[3] and is now being standardized under Linux Foundation[4] with the active involvement of the automotive industry. We discuss more details of Uptane in Section 5.1.

In addition to the security framework by design, engineering efforts have been ongoing too. The International Organization for Standardization (ISO) has been making the software update engineering standard, which is called *ISO 24089: Road Vehicles Software Update*, specifying requirements and recommendations for software update engineering for road vehicles on the organizational and the project level, and vehicles, vehicle systems, infrastructure, and the assembly and deployment of software update packages after the initial development.

3.4 Cybersecurity engineering in automotive

In order to provide an adequate level of security with reasonable efforts, it is important to design and implement systems deploying multi-layered protection based on appropriate threat analysis and risk assessment over the life cycle. In the early stage, SAE introduced the first cybersecurity standard for automotive, SAE J3061 (Society of Automotive Engineers International, 2016), to provide guidance on the security engineering process for securely developing the automotive. As the automotive industry had begun to understand the importance of the cybersecurity engineering process, which was introduced as ISO/SAE 21434 (ISO, 2021). Currently, ISO/SAE 21434 is being adopted as the standard of cybersecurity engineering process for the automotive.

3.5 Standardization and regulation for automotive security

There have been efforts to issue standards and regulations to employ cybersecurity in the automotive. As mentioned in prior sections, several groups, such as AUTOSAR, SAE, and ISO, have been developing cybersecurity standards for various areas, including communication security, system security, and security engineering.

NHTSA/DoT is mandating SCMS for secure V2X communication, and there have been international efforts to regulate automotive security. The United Nations Economic Commission for Europe (UNECE) World Forum for Harmonization of Vehicle Regulations (WP.29) has recently introduced two regulations, Regulation number 155 (UN R155), about cybersecurity and cybersecurity management system (CSMS) (UNECE, 2021a) and Regulation number 156 (UN R156), with regard to software updates and software updates management systems (SUMS) (UNECE, 2021b). For many countries, starting July 2024, it is expected that all new vehicles produced shall comply with these regulations. Each nation adopts UNECE regulations into domestic regulations to implement security in automotive engineering and software updates. UN R155 is aligned with the cybersecurity engineering process standard, ISO/SAE 21434, whereas R156 is aligned with the software update engineering standard, ISO 24089, respectively.

4 Ideas to improve security of UAVs

In this section, we discuss the efforts to improve the security of UAVs. We first show the state of the art of UAV security research and standardization efforts in Section 4.1 and then discuss how automotive security could be deployed into UAVs in Section 4.2.

3 IEEE-ISTO 6100.1.0.0: Uptane Standard for Design and Implementation.

4 https://uptane.github.io/papers/uptane-standard.2.0.0.html.

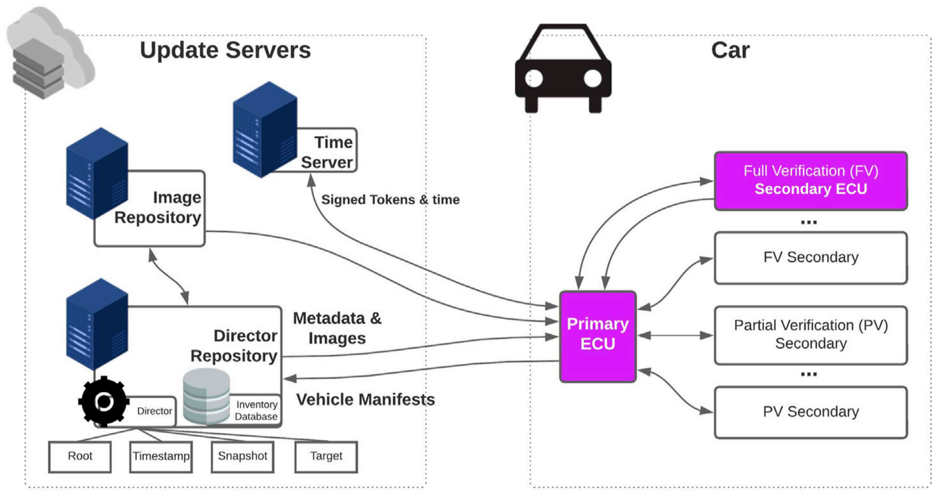

FIGURE 8
Components and update workflow in Uptane Community (2022).

TABLE 4 Challenges in UAV security (Wang et al., 2021) and related categories.

Challenge	Category
UAVs with new radio techniques	Physical security
UAV security in the space–air–ground integrated network	Communication security
Privacy in UAV-enabled applications	Privacy technology
A multi-layer and defense-in-depth security framework	System security, security engineering
UAV-involved communication security standards	Standard and regulation

4.1 State of the art of UAV security

As UAVs are systems with wireless communications, there has been a concern about potential attacks on UAVs. Wang et al. (2021) and Chamola et al. (2021) identified various types of attacks on UAVs in the physical and network layers during wireless communication, including jamming, spoofing, and even spreading malware. Wang et al. (2021) also defined the security requirements of UAVs: data confidentiality, access authentication, system availability, information integrity, and behavior reliability. They also suggested several countermeasures preventing physical attacks, such as jamming and spoofing, as well as encryption and authentication of MAVLink. Moreover, Wang et al. (2021) identified several open issues in securing UAVs, and we map these issues into related categories as in Table 4.

As we categorize the security challenges and countermeasures as physical security, communication security, privacy engineering, system security, security engineering, and standard and regulation, we review how UAV security research and efforts are ongoing.

Physical security has been mostly considered in UAV environments as the threat is on mobile CPS. Various physical attacks, such as jamming and spoofing on sensors and signals, have been studied.

In addition to several countermeasures briefly suggested by Wang et al. (2021), several studies about communication security have focused on providing data confidentiality, information integrity, and access authentication for UAVs. Some researchers (Tanveer et al., 2020; Jan and Khan, 2021; Tanveer et al., 2021; Cho et al., 2020) focused on the establishment of authenticated communication between the ground station and the individual UAV and between UAVs.

Considering the deployment of multiple UAVs as UAV swarms, a few studies have provided secure communication among UAVs. Abdel-Malek et al. (2021) proposed models using proxy signatures over 5G device-to-device (D2D) technologies. In the circumstance that each UAV is capable of cellular communication, each UAV is communicating with other UAVs using 5G D2D technologies.

In certain situations, there could be more constraints, such as the situation in which edge UAVs do not have direct communication with the ground station, whereas each UAV should communicate with other UAVs in the swarm. In such a scenario, the authentication of edge UAVs could happen without the involvement of the ground station. Semal et al. (2018) proposed a certificate-less group authenticated key agreement design that allows communication in a UAV swarm without requiring a certificate for each UAV.

There are also efforts to reduce the burden of the computation of public-key cryptography algorithms for resource-constrained UAV platforms. Khanh et al. (2020) introduced the reputation model that enables immediate decisions on whether the additional authentication process is required and the other additional protocol using it. For the UAV with a higher reputation score, faster authentication is available. However, the leakage of the shared key could result in the malicious entity impersonating the other genuine UAV in the swarm. Han et al. (2022) proposed a model to overcome this limitation and enable lightweight mutual authentication among UAVs, ensuring seamless authentication and resilience against key compromise.

A few works focused on utilizing hardware-assisted methods, such as physically unclonable function- (PUF-) based techniques to enable mutual authentication in UAVs (Pal et al., 2020; Gope and Sikdar, 2020). However, due to the nature of PUF, these protocols require communication between the UAV and the ground station in performing the authentication process.

For Privacy engineering, several researchers (Yao et al., 2017; Tedeschi et al., 2021) focused on privacy-preserving methods as privacy has been a concern for UAV environments. Although the identity of the UAVs is broadcast, the true identities of UAVs are only known to the authorized entity.

Compared to the aforementioned security categories, efforts on system security, security engineering, security standards, and regulations are recently started. There have been requests to develop and introduce security standards and recommendations (ANSI Unmanned Aircraft Systems Standardization Collaborative, 2020) for emerging UAV applications. Several standards committees, such as SAE G-32 Cyber-Physical Systems Security Committee, RTCA SC-216/EUROCAE WG-72 Aeronautical Systems Security, ASTM WK56374, and IETF, develop the security standards for UAVs.

For example, SAE G-32 has started to create the three criteria to help manage risk and ensure the security of CPS throughout their life cycle. SAE G-32 is working on the following standards:

- SAE JA 7496, Cyber-Physical Systems Security Engineering Plan (CPSSEP),
- SAE JA 6678, Cyber-Physical Systems Security Software Assurance,
- SAE JA 6801, Cyber-Physical Systems Security Hardware Assurance.

Although several efforts for standardization and research work and most research focus on communication security and privacy, there are still many open problems for securing the system architecture of UAVs in system security.

4.2 Consideration of boosting UAV system security by employing automotive security

To achieve security over the life cycle of UAVs, providing the multi-layer and defense-in-depth security framework into the system of UAVs, as in Section 2.2.1, is required.

System security focuses on securing UAVs, not only the communication channels but also the internal parts of UAVs. Securing the system includes preserving the system's integrity and data confidentiality, securing internal wired communication, and providing secure software/firmware updates over the life cycle.

As mentioned in Section 2.2.1, a UAV consists of a flight controller, mission computer, and various sensors and motors. These components are interconnected over proprietary connections or internal wired networks, as addressed in Section 2.2.2. This makes the similarity with automotive and employing the security design from the automotive environments reasonable.

4.2.1 Preserving the UAV system's integrity and data confidentiality

As discussed in Section 3.2, deploying HPSE in UAVs is essential to guarantee a secure system to preserve system integrity and data confidentiality.

Although Pirker et al. (2020) focused on deploying HPSE for UAVs, their work remains to provide the HSM as the secure gateway between UAVs and the GCS, and the security of components inside UAVs was barely considered.

It is obvious that the flight controller and the mission computer need to be protected. Employing the HPSE in each component could be the inevitable approach to protect components in the UAV.

Moreover, protecting other independent components, such as motors and sensors installed inside a UAV, should also be considered, as depicted in Figure 3. However, employing the HPSE with equal capabilities to each component may not be easy considering the overall resource constraint including the cost. As discussed in Section 3.2, a vehicle consists of ECUs with different capabilities, and there are three different EVITA grade HSMs and SHE to be employed in each ECU.

Although we do not give what level is sufficient for the UAV component type, the rating of the risk level and the capabilities of the component could be the criteria for the implementer. These decisions should be part of security engineering.

The first criterion is considering the risk. For example, each component may not have an equivalent attack surface. Certain components inside a UAV may be only connected via serial interfaces, such as the inter-integrated circuit (I2C), a universal asynchronous receiver-transmitter (UART), and the serial peripheral interface (SPI). However, some components could be connected over the network, such as CAN, as DroneCAN and Cyphal are specified. How much the component is exposed to the attack surface could be related to how much the chance exists that the component to be targeted by the attacks. It is also interpreted as the opportunity to rate the likelihood in the threat analysis.

The second criterion is the capabilities of components. In the automotive scenarios, if the component handles lightweight and non-critical operations, a lower grade of HPSE could be considered. In contrast, the flight controller or the mission computer of the UAV may need a higher grade of HPSE.

4.2.2 Securing UAV's internal wired communication

Although a few studies have considered securing the MAVLink (Allouch et al., 2019), most researchers focused only on securing the wireless communication between UAVs and between UAVs and the ground station. In contrast, securing communication inside UAVs is barely addressed.

As addressed in Section 2.2.2, the wired communication model in a UAV is similar to the automotive environment. As several studies have focused on securing IVNs, such as CANs and FlexRay, in the automotive domain, adopting the idea into UAVs could be reasonable.

Because the CAN was originally designed without considering the security, 8 bytes of payload in the CAN packet is a huge constraint to providing authentication and encryption. Although CAN-FD allows up to 64 bytes of payload, design efficiency is still required, considering the backward compatibility. Thus, the efforts in securing IVN focused on how to efficiently utilize the constraint payload in the packet, mostly targeted to CAN/CAN-FD.

Therefore, employing the research efforts already performed for automotive in the UAV environment could be easy in supporting security for the UAV communication protocols, such as MAVLink, UranusLink, and especially DroneCAN/Cyphal. For example, DroneCAN/Cyphal is fundamentally the communication protocol running over CAN, and deployment of the secure CAN protocol is immediately considered. Moreover, as explained in Section 2.2.2.1, MAVLink only provides 8 bytes (version 1.0) or 14 bytes (version 2.0) for the payload, which is a similar constraint to CAN.

Additionally, adopting firewalls could be considered to protect communication among the resource constraint devices.

4.2.3 Providing secure software/firmware updates over the UAV's life cycle

The secure software update has been one of the important features in maintaining the system's security during the life cycle, and it is also important for securing UAVs over the life cycle.

Although the commercial UAV market is already providing new software and firmware for their product, how to securely manage the overall software update processes should be considered. Cyberattacks are evolving and various attacks, including malicious software packaging and reverse engineering, could target the UAV software update processes.

Thus, securing the software update processes for UAVs is also important, and a few studies have been introduced. Salamh (2021) showed the threat modeling on UAVs as a service and suggested the use of secure software updates. Al Blooshi and Han (2022) proposed a security model to enable security software updates of UAVs, including UAV swarm environments, adopting Uptane (Karthik et al., 2016; Kuppusamy et al., 2017; Kuppusamy et al., 2018), which is designed to provide a comprehensive security framework for automotive environments.

In Section 5, we show the work of Al Blooshi and Han (2022) as a case study that employing security for automobiles could enable the rapid improvement of the security of UAVs.

5 Case study: Securing UAV software update adopting automotive security

This section shows a case study of employing automotive security technology in UAV environments. Al Blooshi and Han (2022) investigated the similar characteristics between UAVs and the automotive and proposed a security model based on Uptane with principles described in Section 5.1.

5.1 Key principles of Uptane

Uptane is a comprehensive security framework for software updates in the automotive environment, whose scope includes entire ecosystems related to the software update, from infrastructures that manage the software to the target components in the vehicle. Uptane defines roles to separate the duties and authorities and enable multi-layered security on the infrastructure and vehicle sides. By separating roles, the resiliency of the ecosystem against various attacks could be increased, even resiliency when a certain key is compromised via the network breach.

5.1.1 Separation of roles in infrastructures

The main key principle of Uptane for the infrastructure side is the separation of duties, where the responsibility of signing the metadata is distributed between different roles.

As depicted in Figure 8, there are four different basic roles on the repository on the server side: **root, timestamp, snapshot**, and **targets**. Each role is independently managed on the server.

In addition to the roles, Uptane utilizes multiple repositories, known as *director* and *image* repositories, and lets the director repository sign metadata using the root key, whereas the repository is connected to the network while only allowing the image repository to use the root key offline, in which the repository only uses the root key when it is disconnected from the network. This strict policy increases the resiliency that the *root key* at the image repository is secure even when the director repository is compromised.

For more detail on the repositories, refer to Karthik et al. (2016), Kuppusamy et al. (2017), and Kuppusamy et al. (2018).

5.1.2 Primary and secondary ECUs in the vehicle

Uptane categorizes the ECUs in the vehicle into two types: *primary* and *secondary* ECUs. The primary ECU is responsible for distributing information from the repositories to the secondary ECUs and communicating with the different repositories on behalf of all secondary ECUs in a vehicle. This enables the primary ECU to manage the entire ECUs in the vehicle, and it could prevent unauthorized data from flowing to other ECUs. In contrast, the secondary ECU communicates with the primary ECU to receive the new software or report the updated result. The secondary ECU could also verify the software once received. It is considered that primary ECU has more capabilities than secondary ECUs, for example, a TCU in the vehicle.

5.1.3 Metadata and manifest

In order to manage the software, Uptane defines *metadata* and *manifest* to be exchanged between infrastructures and vehicles.

5.1.3.1 Metadata

Metadata are used to provide verifiable information about the software to the vehicle. Uptane defines four types of metadata, and each metadata is generated by each role in the repository. *Root* metadata are generated and signed by the root role, issuing the public keys used to verify metadata created by the other roles. *Timestamp* metadata are generated and signed by the timestamp role, identifying if there are new metadata or images on the repository. *Snapshot* metadata contain version numbers and filenames of the images the repository has released and generated and signed by Snapshot role. *Targets* metadata have information about images, such as hashes and file sizes, and are generated and signed by the Targets role.

When the update is initiated, the repository sends all metadata of the images to the vehicle. The *primary* ECU of the vehicle verifies the image by checking that the hashes and sizes of the signed metadata match the metadata from the repository. The primary ECU sends the metadata and the software images to the secondary ECUs only if the images and metadata are verified. Secondary ECUs also verify the metadata and the image to perform the installation. Upon the scenario, secondary ECUs may perform full verification, which is verifying all metadata, or partial verification, which is verifying only target metadata.

5.1.3.2 Manifest

Manifest is used to provide the vehicle's current software information to the repository. The primary ECU first constructs a *vehicle manifest*, which is a compilation of ECU version reports about the different software versions installed on the ECUs, and sends it to the repository. Using the information in the vehicle manifest and ECU version reports inside, the repository determines which images should be downloaded next. ECU version reports and the vehicle version manifest are sent to the server before the software update and also after the update is completed.

5.2 Challenges in UAV software update

5.2.1 UAV software update scenarios

It is investigated that performing the software update for UAVs could happen in the following two scenarios: performing the software update at the base or during the mission.

5.2.1.1 Software update at the base

Performing software updates at the base is currently being provided for commercial UAVs. Certain vendors already provide the new software and firmware and let UAV users download and install them to UAVs, although security during the software update is not the primary focus.

The scenarios of advanced UAVs with a flight controller and a dedicated mission computer can be considered a simplified model of a vehicle that consists of primary and ECUs. As depicted in Figure 3C, the mission computer communicates with the infrastructure; it could be considered a primary ECU.

Thus, due to many common traits with automotive environments, deploying Uptane for software updates of a UAV is inevitably considered and could be used without any modification.

5.2.1.2 Software update during the mission

For UAV scenarios, there can be scenarios where software updates happen during the mission. In the situation that a critical security update is urgently needed, the software update of the UAV could happen without returning to the base.

Updating UAVs could happen in various scenarios, either a single UAV or multiple UAVs as a group or a drone swarm. Also, certain UAVs could be in situations that cannot perform updates immediately, which will be detailed in Section 5.2.2.

5.2.2 Challenges in UAV environments

Although there is a strong similarity between the UAV and automotive environments, Al Blooshi and Han (2022) identified that there are significant differences between the two, as mentioned in the following sections.

5.2.2.1 Energy constraint in UAV

In the automotive environment, the energy constraint issue is not a critical factor. Instead, the constraint issues were due to computational and communication capabilities in ECUs and IVNs, as well as the complexity of multiple supply chains and the cost of manufacturing the vehicle.

In contrast, the energy constraint issue is one of the most significant issues to limit the utilization of UAVs as battery-powered flying devices.

5.2.2.2 Deploying multiple UAVs as a swarm

In the automotive scenario, grouping multiple vehicles while driving on the same road could happen for platooning. However, although platooning is happening, each vehicle has sufficient capabilities to communicate to the other entities, not only the other vehicle but also the infrastructures, such as a roadside unit.

In contrast, in the UAV scenarios, when multiple UAVs in the cluster could be operated for the same mission, as a "swarm," UAVs with different capabilities could be used. The cluster head (the fog UAV) leading the UAV swarm may have full capabilities to communicate not only with UAVs but also with the GCS, and other UAVs (followers in the swarm) may have capabilities only to communicate within the swarm.

5.2.2.3 UAV as a component and an independent device

As discussed in Section 2.2.1, a UAV may consist of multiple components, including the flight controller and the mission computer. Also, more components could be installed due to the advance in UAV technologies.

Moreover, when multiple UAVs are used as a swarm, it could be the set of UAVs to be installed during the mission. Although all the UAVs are connected in one swarm, they are still considered independent devices.

Although all the components and ECUs in the vehicle perform the updates at the same time, some UAVs may not be able to abort or pause the mission they are performing. For this reason, software updates of the UAV swarm are considered difficult as they may affect the continuity of important missions.

5.2.2.4 Dynamic grouping of the UAV swarm

Although the vehicle consists of a group of many ECUs, they are hardly changed during the lifetime of the vehicle. Conversely, a UAV swarm may consist of different cluster heads (fog UAV) following UAVs in each mission. In addition, UAVs in a swarm during a mission could be changed by joining or leaving the swarm.

This dynamic group makes it hard to use the same mechanism solely as defined in Uptane. Each UAV may generate a version manifest collecting version report from the internal components. The cluster head collects all the version manifests from each UAV because the cluster head is the only entity to interconnect UAV to infrastructure. Moreover, the cluster head needs to manage the UAVs in the swarm.

Thus, a management model for the cluster head to control the software update of the swarm is needed.

5.2.2.5 Wireless communication among UAVs

In the automotive scenario, the update is first delivered from the server to the primary ECU over wireless communication. However, once the update is delivered to the primary ECU, the distribution of the update to the secondary ECUs is not challenging as it is performed over a wired communication, such as CAN.

However, in the case of a UAV swarm, in addition to the communication between the update server and the cluster head, the communications among UAVs are carried out over the wireless channel. As a result, all communication channels within the UAV swarm should consider the attacks targeting the wireless channels.

Security mechanisms for a software update for UAVs satisfying the same security strength of Uptane need customization due to the unique characteristics of the UAV environment.

5.3 Customizing Uptane for UAV environments

5.3.1 Customized manifests

Al Blooshi and Han (2022) introduced the swarm version manifest (SVM), in addition to the drone version manifest (DVM) and component version manifest, to overcome the limitation of Uptane (Kuppusamy et al., 2018) discussed previously.

5.3.1.1 Component version manifest

The component version manifest **(CVM)** is a renamed ECU version report of Uptane and includes the component software status information. In other words, the infrastructure can determine whether the software is up-to-date or needs an update.

The CVM's payload contains the component unique identifier (CUID), information about the installed image, and the time the manifest was generated.

5.3.1.2 Drone version manifest

The **DVM** is renamed the vehicle version manifest in Uptane.

In a single UAV scenario, the DVM is sent to the software update server directly, which is equivalent to Uptane. In contrast, in the swarm scenario, each UAV sends the DVM to the cluster head.

The DVM's payload contains the UAV's unique identifier (DUID), information on the installed image, the time the manifest was generated, and a list of the component version manifests.

5.3.1.3 Swarm version manifest

The **SVM** is defined to customize Uptane for UAV swarm scenarios. By collecting all DVMs, the cluster head constructs the **SVM** and acts as a master list of all the images running on all UAVs or components in the swarm.

The SVM is first generated when the UAV swarm is established and sent to the software update server to let it recognize the dynamic group, as discussed in Section 5.2.2.4. The SVM's payload contains the swarm's unique identifier (SUID), the cluster head's unique identifier (CHUID), and a list of DVMs.

Figure 9 depicts the hierarchy of SVM, DVM, and CVM.

5.3.2 Customized metadata

For the UAV swarm scenarios, *swarm-snapshot* metadata are added, which is the dynamically generated metadata, in addition to the metadata as explained in Section 5.1.3.1. Therefore, the cluster head and the software update server can distinguish the UAVs in the UAV swarm.

5.3.3 Partial postponing update in UAV swarm scenarios

In UAV swarm scenarios, some UAVs may not be able to perform an immediate software update, as discussed in Section 5.2.2.3. Thus, the concept of *postponed* is added as part of the results of the update, in addition to *success* and *failure* in Uptane. The UAVs that cannot perform the immediate update respond with *postponed* in the DVM to the cluster head, and the results are reported to the server. Once they can perform the software update, they perform the software update and report the result.

5.4 Design flow

Upon analysis, as discussed in Section 5.2.2, Al Blooshi and Han (2022) proposed a framework for secure software updates for a UAV swarm environment customizing Uptane. We describe how the Uptane is customized for the UAV environment.

5.4.1 System components

The design by Al Blooshi and Han (2022) defined the components as follows.

5.4.1.1 UAVs

As depicted in Figure 3, a single UAV consists of components, such as a *flight controller* and a *mission computer*, similar to the vehicle, as the vehicle consists of multiple ECUs.

For the UAV swarm case, the UAV is defined as follows:

- **Cluster head (CH)**: It manages the UAV swarm, which consists of one or multiple following UAVs. Only the cluster head communicates with the software update server and the ground station
- **Following UAV (FD):** It is a member of the UAV swarm and controlled by CH. FD may only communicate with other UAVs in most cases.

5.4.1.2 Infrastructures

- **Software update server (SUS):** It distributes the updates to the cluster head in the UAV swarm. The SUS contains all roles and repositories and delivers new updates to the UAVs.

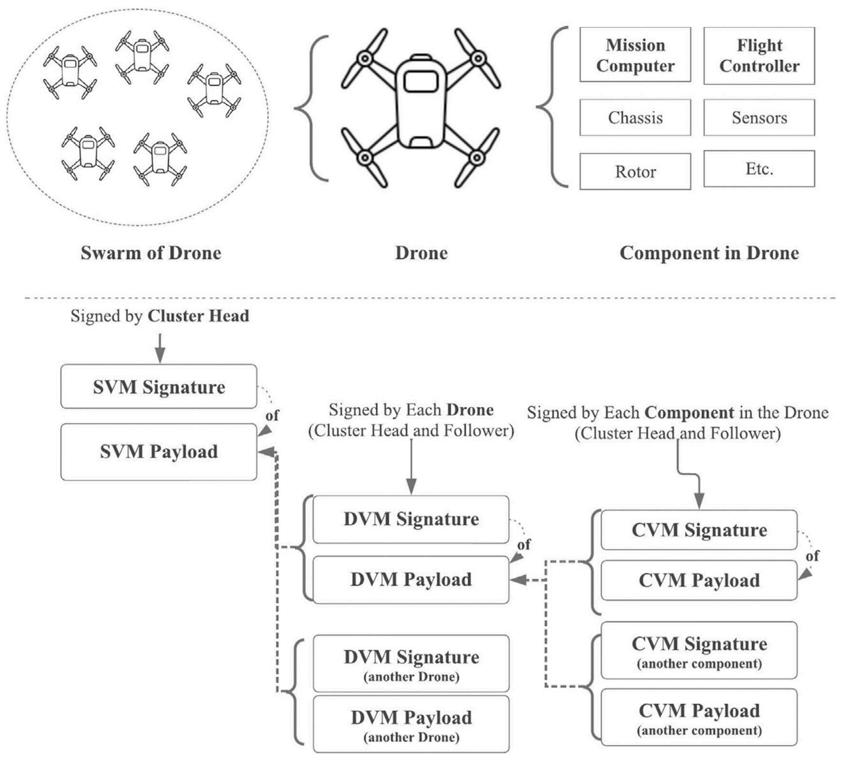
Mission Computer
Flight Controller
Chassis
Sensors
Rotor
Etc.
Swarm of Drone
Drone
Component in Drone
Signed by Cluster Head
SVM Signature
SVM Payload
of
Signed by Each Drone (Cluster Head and Follower)
Signed by Each Component in the Drone (Cluster Head and Follower)
DVM Signature
DVM Payload
of
CVM Signature
CVM Payload
of
DVM Signature (another Drone)
DVM Payload (another Drone)
CVM Signature (another component)
CVM Payload (another component)

FIGURE 9
Swarm version metadata, drone version metadata, and component version metadata. Reproduced from Al Blooshi and Han (2022), with permission from IEEE.

- **Ground control station (GCS):** It is a UAV operator that knows the public keys of UAVs. The GCS is responsible for defining the URL of the update server, uploading the updates to the update server, and establishing a UAV swarm.

5.4.2 Software update phases

The overall software update processes are distinguished into the following three phases:

- **Phase 1 (preparation):** UAVs and servers initially exchange the necessary information to perform the software update (Section 5.4.2.1).
- **Phase 2 (update):** UAVs receive updates from the server and perform the software update (Section 5.4.2.2).
- **Phase 3 (report):** UAVs report the result to the server when the update processes are completed (Section 5.4.2.3).

The detailed flows of each phase in Al Blooshi and Han (2022) are described in Sections 5.4.2.1–5.4.2.3.

5.4.2.1 Phase 1: Preparation

In the initial preparation phase, all relevant entities must be prepared to be able to perform the software update. This stage includes preparation in GCS, SUS, and UAV. If a UAV swarm is established for the mission, GCS generates the UAV swarm DS.

5.4.2.1.1 Swarm preparation in ground station. When the mission requires the UAV swarm DS, the GCS initially establishes DS that includes at least one CH and n number of following UAVs FD_i, where $1 \leq i \leq n$. Establishing DS, the address (*i.e.*, URL) of SUS, and the information about the edge UAV in DS, including UAVs' IDs and public keys, are given to CH at this point. After DS is established, the ground station provides information about DS, $INFO_{DS}$, to the SUS. It is assumed that GCS and SUS have a secure communication channel. $INFO_{DS}$ contains the IDs and associated public keys of all UAVs in DS.

5.4.2.1.2 Preparation in UAV swarm. Establishing the swarm DS, the CH first requests the UAV information from each FD_i in DS, where $1 \leq i \leq n$ (n is the number of UAVs).

- **PD-1**: Each FD_i constructs and sends DVM_i to the CH.
- **PD-2**: CH verifies DVM_i with each FD_i's public key.

Every manifest is signed by its generator, and the signature is included in the manifest, as reported by Kuppusamy et al. (2018).

- **PD-3** CH then constructs and sends SVM[1] to the SUS.

For the single UAV mission, the UAV only constructs DVM and sends it to SUS.

The overall preparation flows are depicted in Figure 10.

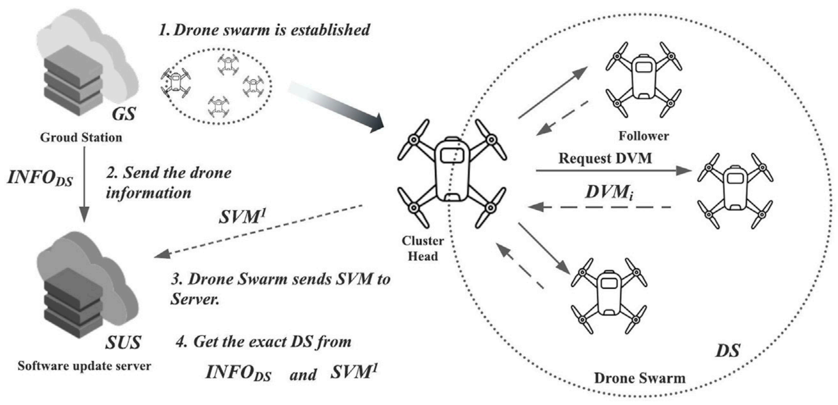
1. Drone swarm is established
GS
Groud Station
$INFO_{DS}$
2. Send the drone information
SVM^1
3. Drone Swarm sends SVM to Server.
SUS
Software update server
4. Get the exact DS from $INFO_{DS}$ and SVM^1
Cluster Head
Follower
Request DVM
DVM_i
DS
Drone Swarm

FIGURE 10
Preparation for the UAV swarm software update. Reproduced from Al Blooshi and Han (2022), with permission from IEEE.

5.4.2.1.3 Preparation in software update server. The SUS contains roles and repositories as defined by Kuppusamy et al. (2018).

In receiving $INFO_{DS}$ from GCS as in Section 5.4.2.1.1 and SVM^1 from the CH as in Section 5.4.2.1.2, the SUS performs the following:

- **PU-1:** The SUS verifies the $INFO_{DS}$ from the GCS and SVM^1 from the CH.
- **PU-2:** The SUS compares $INFO_{DS}$ and SVM^1 to identify the exact members of DS.

Note that the number of UAVs in SVM^1 never exceeds that in $INFO_{DS}$. In contrast, the smaller number of UAVs in SVM^1 could happen due to various reasons, such as weather conditions.

Using the information of DS, the SUS generates *swarm-snapshot* metadata in the update phase.

5.4.2.2 Phase 2: Update

The update phase contains three stages: update initiation (UI), software verification (SV), and installation (IS).

5.4.2.2.1 Update Initiation. When a new update of the software version is issued, the SUS first checks the UAV swarm information collected from SVM^1. If any UAV or components in the UAV swarm DS are considered a target for a software update, the SUS initiates the software update plan. It is assumed that the transmission between the SUS and the CH is over a protected channel such as transport layer security (TLS).

- **UI-1**: The SUS informs the update existence to the CH.
- **UI-2**: The CH constructs and sends *j*th manifest, SVM^j, as described in Section 5.4.2.1.2.

For the first update case, SVM^j may be equivalent to SVM^1. In this case, CH may only send the hash of contents in SVM^1 instead of sending the whole SVM^1 again.

- **UI-3**: The SUS compares SVM^j with SVM^{j-1}.

If the UAV swarm needs an update,

- **UI-4**: The SUS generates the metadata for the update.

For the single UAV mission, *swarm-snapshot* metadata are not necessary.

- **UI-5**: The SUS sends ALL metadata to the CH.

5.4.2.2.2 Software Verification. In receiving the metadata from the SUS, the CH performs the following:

- **SV-1**: CH verifies ALL metadata like Kuppusamy et al. (2018), including *swarm-snapshot*.

CH downloads the software only when all metadata are successfully verified.

- **SV-2**: CH verifies the software images.

If the software image is encrypted, CH only verifies the images in an encrypted state, the same as Uptane.

- **SV-3**: CH sends metadata to each target, where the target is the components in the CH and FDs.

The communication between the CH and the FD is carried out here over a wireless channel. The channel is assumed to be protected using any existing method (*e.g.*, TLS). In receiving metadata from CH, FD performs the following:

- **SV-4**: FD verifies all metadata, and when the metadata are valid, FD downloads software images from the CH or directly from the SUS.

The FD that can perform the update may directly download the software image through SUS using the URL, which is performed to reduce the energy consumption of the FD. In this case, FD can skip downloading and verifying the software images for such UAVs in steps **SV-2** and **SV-3**.

- **SV-5**: The FD verifies the image with metadata (*TARGET* as in Uptane).

After step **SV-5**, the UAV turns to the installation mode.

5.4.2.2.3 Installation. When the software image is validated, each target is ready to install the software images. Once the installation is completed, each UAV moves to the reporting phase. However, UAVs that cannot perform the installation immediately, as explained in Section 5.3.3, may postpone the update and move to the reporting phase without installing the update.

5.4.2.3 Phase 3: Report

In the reporting phase, each component in a UAV would have one of the following results: *success*, *failure*, or *postponed*.

Each component in FD constructs CVM and sends it to FD. In collecting CVM from internal components, FD performs the following:

- **RT-1:** Each FD_i constructs DVM_i and signs it with FD_i's private key.

Although the DVM_i is constructed by collecting new CVM from the internal components, the *postponed* result does not require a new CVM for constructing DVM.

- **RT-2:** Each FD_i sends DVM_i to the CH.
- **RT-3:** The CH constructs SVM^{j+1} with all collected DVM_i and sends it to the SUS.

Results of either *success* or *postponed* in DVM_is are considered completed software updates. However, SVM^{j+1}, including a postponed update, requires the resume of the update when available.

5.4.2.4 Managing postponed software update

If any drone version manifest DVM_i has the result of *postponed*, the CH may check the update applicability of the FD periodically or by following predefined policies.

Upon the predefined policy, the CH may check the status of postponed FDs if they can perform the update. If an FD_i is still unable to perform the update, it shall reply with *postponed* with the detailed reason in DVM_i. If available, the FD_i resumes the update from step **SV-5** again. When the software installation is completed, FD_i constructs and sends the DVM_i with *success* to CH.

As depicted in Figure 11, in step **RT-3**, CH constructs SVM^{j+2} only containing DVM_is from the newly updated FD_is and hash of all up-to-date DVMs instead of all DVMs. Thus, the CH reduces the size of the SVMs when handling the postponed update.

5.5 Evaluation

The proposed design by Al Blooshi and Han (2022) evaluated whether the security model provides security and efficiency while preserving the design fundamentals of Uptane.

5.5.1 Security evaluation of design

5.5.1.1 Security of swarm version manifest

As explained in Section 5.2.2.4, the **SVM** is introduced to support the dynamic grouping, in addition to the renamed **DVM** and **CVM** that are static per drone, the same as the vehicle case.

Security evaluation of DVM and CVM follows Karthik et al. (2016) as they are identical, and this section focuses on the security analysis of the SVM. Let the attacker *Adv* attempt to compromise the SVM. As each manifest is signed by each owner, as depicted in Figure 9, *Adv*, as a third-party attacker, may fail to modify the existing SVM, as well as CVM and DVM, as in Karthik et al. (2016).

Adv may even try to modify SVM after compromising FD. It may include the fraudulent ED^{Adv} into the compromised SVM SVM^{Adv}, as the Adv can generate SVM^{Adv} with a valid signature using the private key of FD. However, the SUS detects the ED^{Adv} by comparing SVM^{Adv} to $INFO_{DS}$ from GS, as ED^{Adv} is not listed in $INFO_{DS}$.

In contrast, *Adv* may exclude DVM from SVM^{Adv} to make the SUS perceive incorrect members in the drone swarm. However, this attack is only limitedly available when *Adv* generates SVM^{Adv} for the first time during the swarm establishment, as SUS checks the number of DVMs with the previous SVMs during the mission.

Moreover, a significant reduction in DVMs may be detected by SUS and GS as they impact the mission. In contrast, excluding a small number of DVMs will be meaningless to operate the mission, as hundreds or thousands of drones are deployed in large-scale missions. To succeed in a meaningful attack, *Adv* has to compromise FD before the preparation in Section 5.4.2.1.2 is completed and exclude enough DVMs that impact the mission without being detected.

For the metadata, although *swarm-snapshot* metadata are included, the security model still follows Karthik et al. (2016).

5.5.1.2 Security over wireless connection in drone swarm

Since FD and ED are connected over a wireless channel, the *Adv* may also perform *Eavesdrop attack*, *Drop-request attack*, *Slow retrieval attack*, and *Endless data attack* between the FD and ED.

5.5.1.2.1 Resiliency against *Eavesdrop attack*. As the metadata and manifests are all signed, no additional security method other than encrypting the channel is necessary. Encryption of the channel can be achieved by any existing methods, such as TLS, as addressed in Section 5.4.2.2.

5.5.1.2.2 Resiliency against *Drop-request attack*. The SVM allows FD and SUS to know the number of drones in the drone swarm by collecting DVMs from each drone.

5.5.1.2.3 Resiliency against *Slow retrieval attack*. Similar to UPTANE (Karthik et al., 2016), all drones can detect and respond to an attack by monitoring the download speed of image metadata and image binaries.

5.5.1.2.4 Resiliency against *Endless data attack*. When downloading the image from the FD or SUS, the edge drone checks the size of the software image in the CVM and DVM. In addition, the FD verifies the size of the images in the CVM, DVM, and SVM.

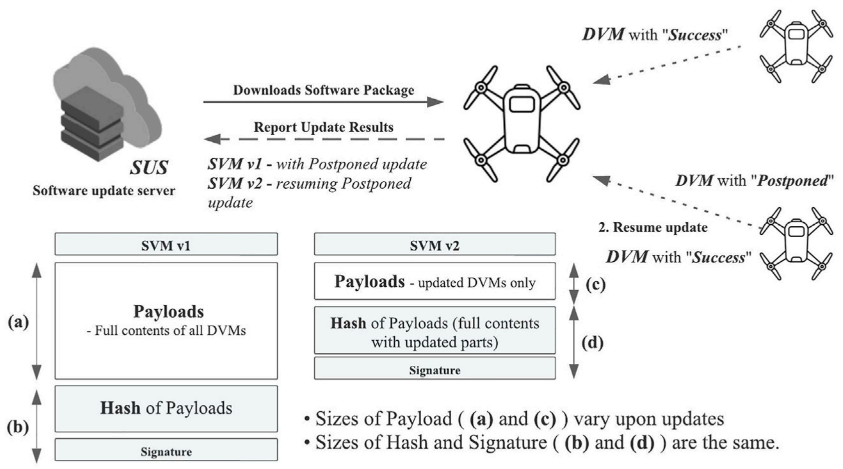
DVM with "Success"
Downloads Software Package
Report Update Results
SUS
Software update server
SVM v1 - with Postponed update
SVM v2 - resuming Postponed update
DVM with "Postponed"
2. Resume update
DVM with "Success"
SVM v1
SVM v2
Payloads - Full contents of all DVMs
Payloads - updated DVMs only
Hash of Payloads (full contents with updated parts)
Hash of Payloads
Signature
(a)
(b)
(c)
(d)
• Sizes of Payload ((a) and (c)) vary upon updates
• Sizes of Hash and Signature ((b) and (d)) are the same.

FIGURE 11
Software update scenario. Handling the UAV's postponed update. Reproduced from Al Blooshi and Han (2022), with permission from IEEE.

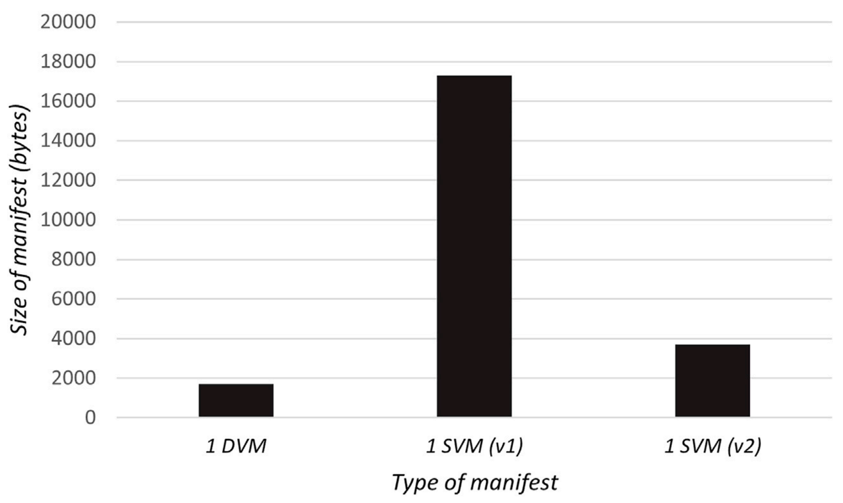

FIGURE 12
Comparison of size of manifests in 1) single UAV update, 2) initial UAV swarm update (v1), and 3) resuming postponed UAV swarm update for two UAVs. Tested with 10 UAVs using Raspberry Pi 4. Reproduced from Al Blooshi and Han (2022), with permission from IEEE.

5.5.1.3 Managing postponed update

Al Blooshi and Han (2022) showed that the design supports the case where some drones may be unable to perform the update while the software update to the drone swarm is initiated, as in Section 5.3.3. FD and SUS can manage the postponed update by checking the version number in SVM.

5.5.1.4 Resiliency against other known attacks

Al Blooshi and Han (2022) showed that the proposed adaptations to Uptane allow us to address specific attacks targeting UAV environments, as described in Section 5.2.2.

As every component in the drone environment verifies the metadata, *Rollback attack*, *Fast Forward attack*, *Partial bundle installation attack*, *Arbitrary software attack*, *Mix-and-match attack*, and *Freeze attack* are mitigated, as shown in Karthik et al. (2016).

Other attacks targeting the wireless communications in the drone swarm, including *Eavesdrop attack*, *Drop-request attack*, *Slow retrieval attack*, and *Endless data* attack, are also mitigated, as discussed in Section 5.5.1.2.

Thus, the customized Uptane for secure UAV software update design remains resilient against the known attacks addressed by Kuppusamy et al. (2018).

5.5.2 Efficiency of design

Al Blooshi and Han (2022) showed the efficiency of the design, which is important in drone environments discussed in Section 5.2.2.1. The proposed design provides an efficient way of sending SVMs, as depicted in Figure 11. Although the first report (SVM v1) after the initial software update has the whole data, the second report (SVM v2) contains only the changed information. The efficiency of the proposed design is also shown by performing a simulation, as depicted in Figure 12. Postponed update significantly reduces the overhead compared to the initial update.

6 Conclusion

While the importance of the security of UAVs is increasing as the applications and architecture of UAVs are becoming more sophisticated, there are still many open security issues to be mitigated in UAV environments, although many studies and standardization activities are ongoing. In this paper, we showed how the efforts to improve automotive security could be deployed in UAV environments, as well as the case study of employing the automotive software update security technology in UAV environments. The results show that automotive security could be deployed in UAV environments with modification while still enabling rapid adoption, preserving the security strength. We believe that employing automotive security technology in UAVs could boost the rapid improvement of the security of UAVs as automotive environments, UAVs have many architectural similarities, and more case studies remain as our future work.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Conflict of interest

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Abdel-Malek, M. A., Akkaya, K., Bhuyan, A., and Ibrahim, A. S. (2021). "A proxy Signature-Based drone authentication in 5G D2D networks," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) (ieeexplore.ieee.org)*, 1–7.

Al Blooshi, S., and Han, K. (2022). "A study on employing UPTANE for secure software update OTA in drone environments," in *2022 IEEE international conference on omni-layer intelligent systems (COINS)*, 1–6.

Aliwa, E., Rana, O., Perera, C., and Burnap, P. (2021). Cyberattacks and countermeasures for in-vehicle networks. *ACM Comput. Surv.* 54, 1–37. doi:10.1145/3431233

Allouch, A., Cheikhrouhou, O., Koubaa, A., Khalgui, M., and Abbes, T. (2019). *MAVSec: Securing the MAVLink protocol for Ardupilot/PX4 unmanned aerial systems.*

ANSI Unmanned Aircraft Systems Standardization Collaborative (2020). STANDARDIZATION ROADMAP for unmanned aircraft systems, version 2.0. *Tech. Rep*. American National Standards Institute.

AUTOSAR (2020). *Specification of secure onboard communication protocol.* 969 Tech. Rep, R20/11, AUTOSAR.

Burgués, J., Hernández, V., Lilienthal, A. J., and Marco, S. (2019). Smelling nano aerial vehicle for gas source localization and mapping. *Sensors* 19, 478. doi:10.3390/s19030478

Chamola, V., Kotesh, P., Agarwal, A., NarenGupta, N., and Guizani, M. (2021). A comprehensive review of unmanned aerial vehicle attacks and neutralization techniques. *Ad Hoc Netw.* 111, 102324. doi:10.1016/j.adhoc.2020.102324

Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., et al. (2011). "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX security symposium*, 77–92.

Cho, G., Cho, J., Hyun, S., and Kim, H. (2020). Sentinel: A secure and efficient authentication framework for unmanned aerial vehicles. *NATO Adv. Sci. Inst. Ser. E Appl. Sci.* 10, 3149. doi:10.3390/app10093149

Ghamari, M., Rangel, P., Mehrubeoglu, M., Tewolde, G. S., and Sherratt, R. S. (2022). Unmanned aerial vehicle communications for civil applications: A review. *IEEE Access* 10, 102492–102531. doi:10.1109/access.2022.3208571

Ghosal, A., and Conti, M. (2020). Security issues and challenges in V2X: A survey. *Comput. Netw.* 169, 107093. doi:10.1016/j.comnet.2019.107093

Gope, P., and Sikdar, B. (2020). An efficient privacy-preserving authenticated key agreement scheme for edge-assisted internet of drones. *IEEE Trans. Veh. Technol.* 69, 13621–13630. doi:10.1109/TVT.2020.3018778

Han, K., Nuaimi, E. A., Blooshi, S. A., Psiakis, R., and Yeun, C. Y. (2022). "A new scalable mutual authentication in fog-edge drone swarm environment," in *Information security practice and experience*. Editors C. Su, D. Gritzalis, and V. Piuri (Cham: Springer International Publishing), 179–196.

Han, K., Weimerskirch, A., Shin, K. G., Kim, Y., Lee, S. E., et al. (2014). Characteristics of hematologic malignancies with coexisting t(9;22) and inv(16) chromosomal abnormalities. *IQT Q.* 6, 22–28. doi:10.5045/br.2014.49.1.22

Henniger, O., Ruddle, A., Seudié, H., Weyl, B., Wolf, M., and Wollinger, T. (2009). "Securing vehicular on-board it systems: The evita project," in *VDI/VW automotive security conference*, 41.

Intelligent Transportation Systems Joint Program Office. Security credential management system proof of concept, the US department of transportation. Available at: https://www.its.dot.gov/resources/scms.htm

ISO (2021). ISO/SAE 21434: 2021: Road vehicles: Cybersecurity engineering *(ISO).*

Jan, S. U., and Khan, H. U. (2021). Identity and aggregate signature-based authentication protocol for IoD deployment military drone. *IEEE Access* 9, 130247–130263. doi:10.1109/access.2021.3110804

Karthik, T., Brown, A., and Kuppusamy, S. A. (2016). "Uptane: Securing software updates for automobiles," in *International conference on embedded security in car. 2016.*

Kenjic, D., and Antic, M. (2022). Connectivity challenges in automotive solutions. *IEEE Consum. Electron. Mag.* 1–6, 1–6. doi:10.1109/mce.2022.3183807

Khan, N. A., Jhanjhi, N. Z., Brohi, S. N., and Nayyar, A. (2020). "Chapter three - emerging use of uav's: Secure communication protocol issues and challenges," in *Drones in smart-cities*. Editor F. Al-Turjman (Elsevier), 37–55. doi:10.1016/B978-0-12-819972-5.00003-3

Khanh, T. D., Komarov, I., Don, L. D., Iureva, R., and Chuprov, S. (2020). "Tra: Effective authentication mechanism for swarms of unmanned aerial vehicles," in *2020 IEEE symposium series on computational intelligence (SSCI)*, 1852–1858.

Kriz, V., and Gabrlik, P. (2015). UranusLink - communication protocol for UAV with small overhead and encryption ability. *IFAC-PapersOnLine* 48, 474–479. doi:10.1016/j.ifacol.2015.07.080

Kuppusamy, T. K., DeLong, L. A., and Cappos, J. (2017). Securing software updates for automotives using uptane. *login Usenix Mag.* 42.

Kuppusamy, T. K., DeLong, L. A., and Cappos, J. (2018). Uptane: Security and customizability of software updates for vehicles. *IEEE Veh. Technol. Mag.* 13, 66–73. doi:10.1109/mvt.2017.2778751

Lee, J.-H., Lee, J., and Cha, J. (2018). "How to build controller area network communication test environment using NVIDIA TX2 for unmanned aerial vehicle," in *2018* 3rd international Conference on Smart and sustainable technologies (SpliTech), 1–3.

Matsumoto, T., Hata, M., Tanabe, M., Yoshioka, K., and Oishi, K. (2012). "A method of preventing unauthorized data transmission in controller area network," in *IEEE vehicular technology conference*.

Moore, M., McDonald, I., Weimerskirch, A., Awwad, S., DeLong, L. A., and Cappos, J. (2020). *Using a Dual-Layer specification to offer selective interoperability for uptane*.

Müter, M., and Asaj, N. (2011). "Entropy-based anomaly detection for in-vehicle networks," in *IEEE intelligent vehicles symposium, proceedings*, 1110–1115.

Nilsson, D. K., Larson, U. E., and Jonsson, E. (2008). "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *IEEE vehicular technology conference*, 1–5.

Nilsson, D. K., Larson, U. E., Picasso, F., and Jonsson, E. (2009). A first simulation of attacks in the automotive network communications protocol flexray. *Adv. Soft Comput.* 53, 84–91.

Pal, V., Acharya, B. S., Shrivastav, S., Saha, S., Joglekar, A., and Amrutur, B. (2020). "PUF based secure framework for hardware and software security of drones," in *2020 asian hardware oriented security and trust symposium (AsianHOST)*, 01–06.

Pirker, D., Fischer, T., Lesjak, C., and Steger, C. (2020). "Global and secured uav authentication system based on hardware-security," in *2020 8th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)* (Los Alamitos, CA, USA: IEEE Computer Society), 84–89. doi:10.1109/MobileCloud48802.2020.00020

Pothuganti, K., Jariso, M., and Kale, P. (2017). "A review on geo mapping with unmanned aerial vehicles," in *International journal of innovative research in computer and communication engineering*, 3297.

Qureshi, A., Marvi, M., Shamsi, J. A., and Aijaz, A. (2021). *eUF: A framework for detecting over-the-air malicious updates in autonomous vehicles*. Journal of King Saud University - Computer and Information Sciences.

Sae International (2020). *SAE-J3101: Hardware protected security for ground vehicles*. Society of Automotive Engineers International. Tech. rep.

Salamh, F. E. (2021). Security and law security and law a ConstructivA constructive DIREST security thre DIREST security threat modeling for dreat modeling for drone as a one as a SerService vice. *J. Digital Forensics, Secur. Law* 16.

Samuel, J., Mathewson, N., Cappos, J., and Dingledine, R. (2010). "Survivable key compromise in software update systems," in *Proceedings of the ACM conference on computer and communications security*, 61–72.

Schweppe, H., Roudier, Y., Weyl, B., Apvrille, L., and Scheuermann, D. (2011). "Car2X communication: Securing the last meter - a cost-effective approach for ensuring trust in Car2X applications using in-vehicle symmetric cryptography," in *IEEE vehicular technology conference*.

Semal, B., Markantonakis, K., and Akram, R. N. (2018). "A certificateless group authenticated key agreement protocol for secure communication in untrusted UAV networks," in *2018 IEEE/AIAA 37th digital avionics systems conference (DASC)*, 1–8.

Society of Automotive Engineers International (2016). *Sae J3061 - cybersecurity guidebook for cyber-physical vehicle systems*. Tech. rep.

Tanveer, M., Khan, A. U., Kumar, N., and Hassan, M. M. (2021). "RAMP-IoD: A robust authenticated key management protocol for the internet of drones," in *IEEE internet of things journal*, 1.

Tanveer, M., Zahid, A. H., Ahmad, M., Baz, A., and Alhakami, H. (2020). LAKE-IoD: Lightweight authenticated key exchange protocol for the internet of drone environment. *IEEE Access* 8, 155645–155659. doi:10.1109/access.2020.3019367

Tedeschi, P., Sciancalepore, S., and Di Pietro, R. (2021). "Arid: Anonymous remote identification of unmanned aerial vehicles," in *Annual computer security applications conference* (New York, NY, USA: Association for Computing Machinery), 207–218. ACSAC '21. doi:10.1145/3485832.3485834

UNECE (2021a). *UN Regulation No. 155 - cyber security and cyber security management system*.

UNECE (2021b). *UN Regulation No. 156 - software update and software update management system*.

Uptane Community (2022). Uptane standard for design and implementation 2.0.0. Linux Foundation joint Development Foundation Projects, LLC.

Wang, L., Chen, Y., Wang, P., and Yan, Z. (2021). Security threats and countermeasures of unmanned aerial vehicle communications. *IEEE Commun. Stand. Mag.* 5, 41–47. doi:10.1109/mcomstd.0001.2000078

Whyte, W., Weimerskirch, A., Kumar, V., and Hehn, T. (2013). "A security credential management system for V2V communications," in *2013 IEEE vehicular networking conference*, 1–8.

Yao, Y., Xia, H., Huang, Y., and Wang, Y. (2017). "Privacy mechanisms for drones: Perceptions of drone controllers and bystanders," in *Proceedings of the 2017 CHI conference on human factors in computing systems* (New York, NY, USA: Association for Computing Machinery), 6777–6788. CHI '17. doi:10.1145/3025453.3025907