applied sciences

MDPI

Article

SENTINEL: A Secure and Efficient Authentication Framework for Unmanned Aerial Vehicles

Geumhwan Cho [1], Junsung Cho [1], Sangwon Hyun [2] and Hyoungshick Kim [1,3,*]

[1] Security Engineering Lab (27317), Department of Electrical and Computer Engineering, College of Information and Communication Engineering, Sungkyunkwan University (SKKU), Suwon 16419, Korea; geumhwan@skku.edu (G.C.); js.cho@skku.edu (J.C.);
[2] Department of Computer Engineering, Myongji University, Yongin, Gyeonggi-do 17058, Korea; shyun@mju.ac.kr
[3] Data61, CSIRO, Marsfield 2122, Australia
* Correspondence: hyoung@skku.edu

Received: 30 March 2020; Accepted: 28 April 2020; Published: 30 April 2020

Abstract: Extensive use of unmanned aerial vehicles (commonly referred to as a "drone") has posed security and safety challenges. To mitigate security threats caused by flights of unauthorized drones, we present a framework called SENTINEL (**S**ecure and **E**fficient authe**NTI**cation for u**N**manned a**E**rial vehic**L**es) under the Internet of Drones (IoD) infrastructure. SENTINEL is specifically designed to minimize the computational and traffic overheads caused by certificate exchanges and asymmetric cryptography computations that are typically required for authentication protocols. SENTINEL initially generates a flight session key for a drone having a flight plan and registers the flight session key and its flight plan into a centralized database that can be accessed by ground stations. The registered flight session key is then used as the message authentication code key to authenticate the drone by any ground station while the drone is flying. To demonstrate the feasibility of the proposed scheme, we implemented a prototype of SENTINEL with ECDSA, PBKDF2 and HMAC-SHA256. The experiment results demonstrated that the average execution time of the authentication protocol in SENTINEL was about 3.1 times faster than the "TLS for IoT" protocol. We also formally proved the security of SENTINEL using ProVerif that is an automatic cryptographic protocol verifier.

Keywords: unmanned aerial vehicle; authentication; key management; digital certificate; internet of drones

1. Introduction

Unmanned aerial vehicles (UAVs), also referred to as drones, have been primarily used for military purposes such as observing the enemy. Recently, they have used in civilian such as delivery services and entertainment [1,2]. Due to the increase in the use of drones, the threats caused by the drones might also arise. Indeed, there are several security and safety issues raised by drones in the real world. In 2015, a drone crashed on the grounds of the White House [3]. That drone, DJI Phantom FC40, was too small and flying too low to be detected by radar. Although this incident was the result of an unintentional and inadvertent mistake, this happening demonstrates that drones can be used for military operations (e.g., delivering explosives and spying on people and buildings).

To mitigate security risks caused by such (unknown) drones, we need to verify the identities of flying drones in real-time and prevent the flights of unauthorized drones. Recently, the concept of the Internet of Drones (IoD) was introduced as a network architecture to coordinate the access of drones in the airspace. In the IoD environment, each drone is always connected to a *ground station* associated with a fly zone using the Internet connection based on a cellular network such as LTE and

5G network. Figure 1 shows the communication channels between drones and ground stations in the IoD environment. Whenever a drone enters a new fly zone (Fly Zone *j* in Figure 1), the ground station in the fly zone checks whether the drone is authorized to fly in that zone using a security protocol. We assume that a 5G network connection is provided for the communication between the drone and the ground station. Based on the 5G network, a secure communication channel can be established between them. However, such a secure communication channel alone is not sufficient to detect flights of unauthorized drones that are capable of using the 5G network. To detect them, we need to additionally check whether a detected drone in a fly zone is authorized or not. That is, an authentication protocol for drones is required. Unlike typical IoT protocols, the protocol in IoD is required to satisfy several requirements, such as real-time authentication and scalability. In IoD, numerous drones would be operated simultaneously. Therefore, the authentication protocol for IoD should be scalable to be able to cope with the increasing number of drones. However, in several existing IoT protocols (e.g., [4–6]), a trusted third party is required while performing the authentication process. In this paper, we aim to design a highly scalable protocol to process a massive number of drones concurrently without relying on a single server. Moreover, the ground station should authenticate drones as fast as possible by using a short message and reducing the number of communications for real-time authentication. Therefore, we aim to develop a lightweight authentication protocol using a MAC (message authentication code) scheme with the new concept of the flight session key.

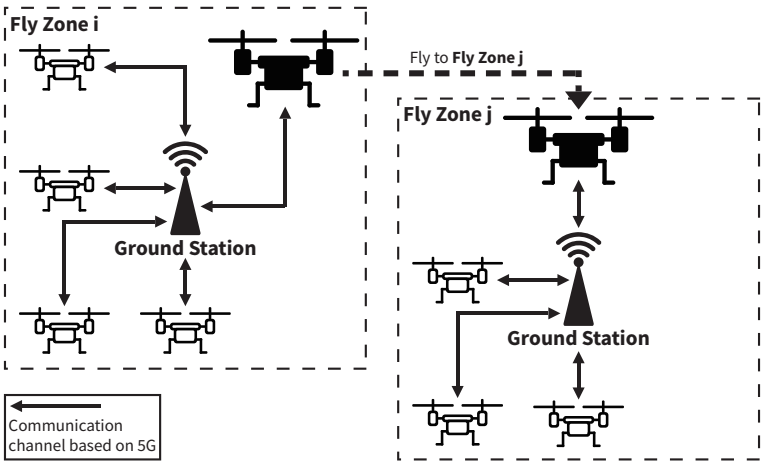


Figure 1. Communication channels between drones and ground stations in IoD.

Many security protocols (e.g., [7–9]) have been developed for authentication. Among them, digital certificate-based authentication mechanisms are more preferably used to authenticate devices and hosts on public communication channels such as the Internet. Notably, many Internet of Things (IoT) standards such as OCF (Open Connectivity Foundation) [10] and IoTivity [11] adopt a certificate-based scheme to authenticate IoT devices and establish secure sessions between them without user intervention. In theory, certificate-based authentication seems a neat solution for unmanned aerial vehicles (i.e., drones). However, to deploy such an existing certificate-based framework in the IoD environment, we need to address a challenging issue – certificate-based schemes are too expensive in resource-constrained drones with respect to communication overhead, power consumption, and memory footprint. First, it takes much time to exchange certificates between devices [12]. Therefore, we should avoid doing the certificate exchange process as many as possible. Second, asymmetric cryptography used in certificate-based schemes is significantly slower and requires more power consumption than symmetric cryptography [13]. Therefore, it seems better to use symmetric cryptography instead of using asymmetric cryptography. Third, since the size of a digital certificate is quite large, it could be a burden for resource-constrained drones. Currently, X.509 v3 [14] is most popularly used as the de facto standard of digital certificates, and its size is about 1 KB–1.5 KB on average [15]. However, because the X.509 v3 certificate was initially designed for

computers and servers with sufficient resources, thus it is desirable to develop a lightweight certificate format that is viable for IoD.

In this paper, we propose an authentication framework named SENTINEL (**S**ecure and **E**fficient authe**NTI**cation for u**N**manned a**E**rial vehic**L**es) for IoD to provide mutual authentication between drones and ground stations. To avoid the computational and traffic overheads caused by certificate exchanges and asymmetric cryptography operations, we introduce the idea about the *flight session key* between a drone and a ground station created by the key agreement protocol between them once the drone takes off. The flight session key is used as a MAC (message authentication code) key for efficiently authenticating drones in subsequent sessions whenever authentication is needed. To provide a secure key agreement protocol, we build public key infrastructure (PKI) using a lightweight certificate format that is more suitable for the IoD environment than the conventional X.509 certificate format.

Our main contributions are summarized as follows:

- We propose SENTINEL (**S**ecure and **E**fficient authe**NTI**cation for u**N**manned a**E**rial vehic**L**es) to provide mutual authentication between drones and ground stations. SENTINEL initially generates a *flight session key* for a drone having a flight plan and registers the flight session key and its flight plan into a centralized database that can be accessed by all ground stations. The registered flight session key is used to authenticate the drone by ground stations as the MAC key while it is flying (see Sections 4 and 5).
- We designed a lightweight certificate format that is suitable for IoD environments. Compared to the conventional X.509 v3 certificate, the proposed certificate is designed to contain only minimal information required to build public key infrastructure in IoD environments. To further reduce the size of certificates, we adopt a binary format in the proposed certificate, instead of a human-readable text format adopted in X.509 v3 certificates. Our results show that the size of the proposed certificate was about 3.7 times smaller than the size of a typical X.509 v3 certificate (see Section 6).
- We implement a prototype of SENTINEL to show its feasibility and efficiency. Our results demonstrate that the execution time of the authentication protocol in SENTINEL is 3.1 times faster than the "TLS for IoT" protocol on average (see Section 7). We also formally verify the security of SENTINEL using ProVeif, which is a formal verification tool (see Section 8).

The rest of this paper is organized as follows. In Section 2, we give an overview of the related work. In Section 3, we provide background knowledge such as terminology, notations, and present the threat model considered in this paper. In Section 4, we describe the overview of SENTINEL. In Section 5, we present the protocols for key generation and authentication. In Section 6, we present the digital certificate structure used in SENTINEL. In Section 7, we present the experimental results to show the efficiency of SENTINEL. In Section 8, we formally verify the proposed framework using a verification tool called ProVerif. Finally, in Section 9, we discuss conclusions.

2. Related Work

2.1. Security and Privacy in IoD

Gharibi et al. [16] proposed the concept of the Internet of Drones (IoD), which is a layered network architecture to coordinate access of drones to controlled airspace and provide navigation services. The proposed architecture is composed of five layers, including airspace, node-to-node, end-to-end, services, and applications, and each layer is responsible for providing its services to the upper layers. Lin et al. [17] discussed security and privacy concerns that should be considered in IoD environments, such as the authentication of drones and ground stations, the identity and location privacy of drones, the confidentiality of data outsourced from drones into the cloud and secure data sharing among collaborating drones, and also proposed solutions to ensure the identity and location privacy of

drones and the confidentiality of outsourced data. Ni et al. [18] also proposed a scheme to ensure the location privacy of each device by encrypting its location information using Elgamal and AES. The authentication framework for drones in [19] provides preserving the privacy of drones' identities and locations. The user authentication approaches in [5,6] also ensure the anonymity of users whose identities are verified by drones.

2.2. Authentication among Drones, Ground Stations, and Other Parties

Besides the drone incident mentioned in Section 1, several other incidents were motivating drone authentication for detecting suspicious/unauthorized drones within fly zones. In July 2018, Greenpeace members intentionally crashed a drone into the nuclear plant facility in Lyon, France, to warn the vulnerabilities of critical infrastructures against attacks [20]. In December 2018, there was another serious incident at London Gatwick Airport, where the airport was shut down for about 30 hours due to the unknown drones around the runway, which resulted in disrupting approximately 1000 flights and 140,000 passengers [21]. In August 2016, there was a malicious attempt using a drone to secretly film a victim entering the PIN and withdrawing money from an ATM to identify the victim's PIN [22] eventually. All these incidents demonstrate that it is critical to detect and disable flights of unauthorized drones.

Several studies focused on designing secure and efficient authentication schemes for drones, ground stations, and other smart objects (e.g., sensors). Won et al. [23] proposed a suite of cryptographic protocols to provide an authenticated key agreement between drones and smart objects in smart city environments to enable secure communications in three different communication scenarios including one-to-one, one-to-many, and many-to-one. The authors mainly focused on designing protocols without relying on public key infrastructure. Tian et al. [19] proposed an authentication framework for secure communication between drones which utilizes an online/offline signature scheme and the assistance of mobile edge computing technology. Besides secure authentication, the proposed approach also considers preserving the privacy of drones' identities and locations. Chen et al. [24] proposed an authentication system for IoD environments to provide mutual authentication among system entities such as drones, operators, and ground stations. To be specific, this approach relies on the coordination of the trusted authority center which issues public/private key pairs for system entities, and a ground station decides whether to approve the flight path of a drone or not through mutual authentication between the drone and the ground station as well as between the operator of the drone and the ground station. Although these approaches also use a simplified public key infrastructure for issuing keys to system entities along with digital signatures, they did not take into account the computational and communication overheads introduced by exchanging keys and signatures and performing signature operations. Unlike these approaches, our approach minimizes the use of expensive certificate-based authentication only for the initial authentication of each drone to generate a flight session key. Drones can use a more lightweight authentication scheme with the flight session key for all subsequent sessions to reduce the authentication overheads. Our approach also uses a lightweight certificate format that is more suitable for resource-constrained drones than the conventional X.509 certificate format.

2.3. Authentication between Users and Drones

Several researchers studied user authentication to allow only authorized users to access data available on remote drones. Srinivas et al. [5] proposed a user authentication mechanism for drones that jointly uses the mobile device, password, and biometric of a user as multiple authentication factors. The proposed authentication mechanism is based on temporary credentials to ensure user anonymity in addition to restricting unauthorized access to drones. Zhang et al. [6] proposed a lightweight cryptographic protocol to provide mutual authentication and key agreement between users and drones by using only cryptographic hash functions and XOR operations. The proposed approach additionally requires the intervention of a control server. Wazid et al. [4] presented a comprehensive review of user

authentication mechanisms to drones in IoD environments. These studies have mainly focused on obtaining the authority of controlling drones through authentication between drones and remote users via ground stations, which are assumed as trusted authorities. Unlike these approaches, the main focus of our work is to check whether a drone in the fly zone obtains the authorization of flight or not through the authentication between drones and ground stations.

3. Background

To provide a better understanding of the proposed framework, SENTINEL, we introduce the terminology and notations used in the remaining sections. We also describe the threat model that we consider in this paper.

3.1. Terminology and Notations

We use the following terms throughout the paper.

- **Drone** is an entity (known as an unmanned aerial vehicle, UAV) designed to be flown either through remote or autonomous controls using embedded software and sensors. In this paper, a drone is recognized as *authorized* one when a ground station successfully verifies the authenticity of the drone.
- **Ground station** is an entity that allows drones to communicate with other drones using a 5G network and has the role of tracking the locations of (authorized) drones during their flight.
- **Certificate authority (CA)** is an entity which performs two roles. The first role is to issue a certificate to drones and ground stations. Another role is to confirm whether the certificate is revoked by comparing the certificate with the certificate revocation list.
- **Operator** is a person or a company that wants to control a drone. It requests a certificate for her drone to CA. After having the certificate, it has to install the certificate on the drone. We assume that each drone has a unique certificate issued by a legitimate CA.

We use the following notations in the description of security protocols in Table 1.

Table 1. Notations used for the protocols in SENTINEL.

Notation	Description
$\|$	Concatenation operation
id_d	Identity of drone d
fp_d	Flight plan for drone d
n_d	Nonce of drone d
n_{gs}	Nonce of ground station gs
$cert_d$	Certificate of drone d
$cert_{gs}$	Certificate of ground station gs
sk_d	Private key of drone d
sk_{gs}	Private key of ground station gs
pk_d	Public key of drone d
pk_{gs}	Public key of ground station gs
pk_{ca}	Public key of CA ca
Sig	Signing function
Enc	Encryption function
Dec	Decryption function
$PBKDF$	Password-based key derivation function
pms	Pre-Master-Secret key
k_d	Flight session key for drone d
k_{gs}	Flight session key for ground station gs
$HMAC$	Hash-based message authentication code function
H_d	Hash value computed by drone d
H'_d	Hash value computed by ground station gs
msg_d	Message of drone d

3.2. Threat Model

We consider an unauthorized drone as an attacker. Based on Dolev and Yao's paper [25], as the threat model, we use a computationally bounded attacker running in polynomial time, and is not capable of breaking cryptographic primitives without knowing the key(s) used for those cryptographic primitives with the following assumptions.

- The attacker can eavesdrop on a public channel. That is, the attacker can steal or drop the message between drones and ground stations.
- The attacker can also send any messages to drones and ground stations.

The attacker's goal is to deceive a ground station into believing that the messages from the attacker are authentic (coming from an authorized drone).

4. Overview of SENTINEL

In this section, we describe an overview of SENTINEL whose objective is to verify the authenticity of drones for detecting unauthorized drones in each fly zone. Figure 2 shows how SENTINEL works with the four entities—drone, ground station, certificate authority (CA), and operator (described in Section 3.1)—to achieve the objective of SENTINEL eventually. First of all, the operator obtains a certificate for the drone he/she owns from the CA (step 'a. Register' and 'b. Certificate') and installs the certificate on the drone (step 'c. Install Certificate'). The ground station also obtains its certificate from the CA (step 'a. Register' and 'b. Certificate'). After these steps, the drone, its operator, and the ground station are all registered to SENTINEL. That is, the drone and the ground station can be identified using the certificates issued by the CA.

Each drone must initially obtain approval of its flight plan about which fly zones it will fly in from SENTINEL before its flight. This approval procedure is carried out based on the two-party mutual authentication and key agreement protocol between the drone and the ground station (For more details of the mutual authentication and key agreement protocol provided by SENTINEL, please refer to Section 5.1). As shown in step 1 in Figure 2, the drone and the ground station performs the mutual authentication and key agreement protocol. During this process, the drone also requests an approval of its flight plan to the ground station. Then the ground station decides whether or not to approve the flight plan requested from the drone. Consequently, a *flight session key* for the drone is uniquely generated through the key agreement protocol between the drone and the ground station. As shown in step 2 in Figure 2, the approved flight plan, along with the associated drone's identity and the flight session key, is stored in the flight information database ('Flight Info. DB' in Figure 2), which is shared by all the ground stations in SENTINEL. By referring to this flight information database, a ground station decides on whether or not to allow the flight of each drone entering its fly zone. That is, when the drone moves from "fly zone i" to "fly zone j", a ground station located in "fly zone j" can identify and authenticate the drone by accessing the flight information database.

Whenever the drone enters a new fly zone during its flight, it performs an authentication protocol with the ground station associated with the new fly zone by using the flight session key (see Section 5.2). For the authentication protocol, the drone first sends a message, including the drone's identity and authenticated code value computed with the flight session key to the ground station. After receiving the message, the ground station retrieves the drone's flight information (the flight session key and flight plan approved for the drone) with the drone's identity from the flight information database. The ground station then verifies the authenticity of the received authenticated code value with the flight session key and decides to allow the drone to fly in the current fly zone by checking whether that fly zone is contained in the approved flight plan.

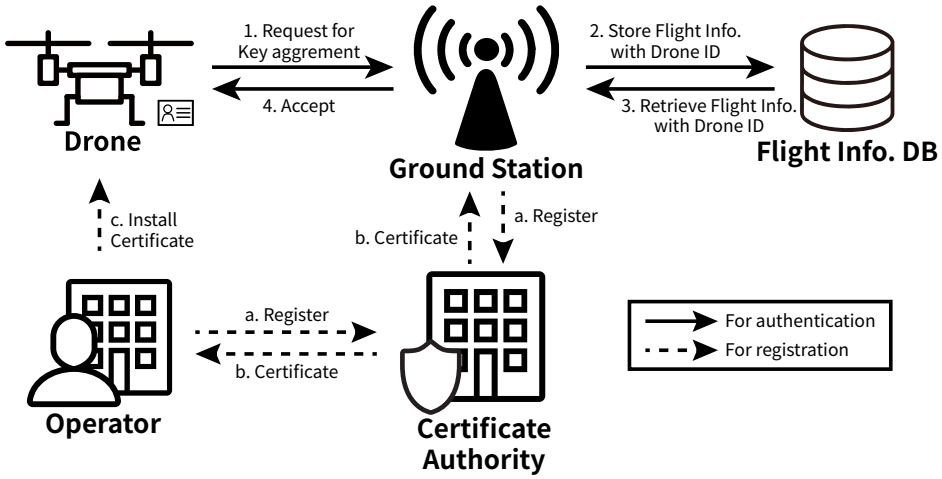


Figure 2. Overall process of SENTINEL.

5. Security Protocols in SENTINEL

In this section, we present the security protocols between drones and ground stations showing to generate a *flight session key* for drone authentication, and how the ground station authenticates the drones with the flight session key.

5.1. Mutual Authentication and Key Agreement Protocol

Figure 3 shows the mutual authentication and key agreement protocol initially performed between a drone and a ground station. We assume that both the drone and the ground station already have their certificates issued by a legitimate CA (described in Figure 2). In this protocol, the ground station checks whether the drone is an authorized drone by verifying the validity of the drone's certificate and generates a flight session key with the drone, which will be used for authenticating the drone during the drone's actual flight.

That is, before taking off, the drone d tries to ask the approval from the ground station gs described in Figure 3 through the mutual authentication and key agreement protocol. During the protocol, d and gs perform mutual authentication with their certificates and verifies each other's certificate by checking the signature on the certificate. Optionally, both the drone and ground station can check whether the certificate is revoked using a certificate validation service such as an online certificate status protocol (OCSP). The details of the procedure for the key generation are as follows.

1. When a drone d tries to start a flight, d first chooses a random nonce n_d using a cryptographically secure pseudorandom function, and sends the message including drone's identity id_d, n_d, drone's certificate $cert_d$, and drone's flight plan fp_d to a predesignated ground station gs.
2. On receiving the message from d, gs also chooses its random nonce n_{gs}. Then, gs signs "n_d and n_{gs}" with the ground station's private key sk_{gs}. We use $Sig(sk_{gs}, n_d, n_{gs})$ to represent the signed message of "n_d and n_{gs}" with sk_{gs}. gs sends the message including the signed value $Sig(sk_{gs}, n_d, n_{gs})$ and the ground station's certificate $cert_{gs}$.
3. After sending the messages, gs checks the validity of $cert_d$ received from d. At the same time, d also checks the validity of $cert_{gs}$. Then, d extracts the ground station's public key pk_{gs} from $cert_{gs}$ to check the validity of $Sig(sk_{gs}, n_d, n_{gs})$. As a result, d has a valid n_{gs} which will be used for generating the flight session key.
4. gs extracts the public key of pk_d from $cert_d$. After extracting pk_d , gs generates a pre-master-secret key pms using a cryptographically secure pseudorandom function and then encrypts pms with pk_d, which resulted in $Enc(pk_d, pms)$. Then, gs signs $Enc(pk_d, pms)$ with sk_{gs} and sends $Sig(sk_{gs}, Enc(pk_d, pms))$ to d.

5. As for the last step, gs concatenates n_d and n_{gs} denoted as $n_d \parallel n_{gs}$. Here, random nonces are used to prevent replay attacks. Then, gs computes a flight session key k_d using a key derivation functions (e.g., PBKDF2 [26] with 10,000 iteration) with pms and $n_d \parallel n_{gs}$ together. At the side of the drone, d first checks the validity of $Sig(sk_{gs}, Enc(pk_d, pms))$ with pk_{gs}. If the signature is valid, then d decrypts $Enc(pk_d, pms)$ with its private key sk_d. Finally, d also computes the flight session key k_d using a key derivation function with pms and $n_d \parallel n_{gs}$, which is the exactly same as the key generation by gs.
6. After completing the generation of the flight session key k_d, gs registers the drone d's identity id_d, flight session key k_d and flight plan fp_d to the flight information database.

If we complete this protocol without errors, gs and d commonly share a flight session key. We will formally verify the security analysis of this protocol in Section 8.

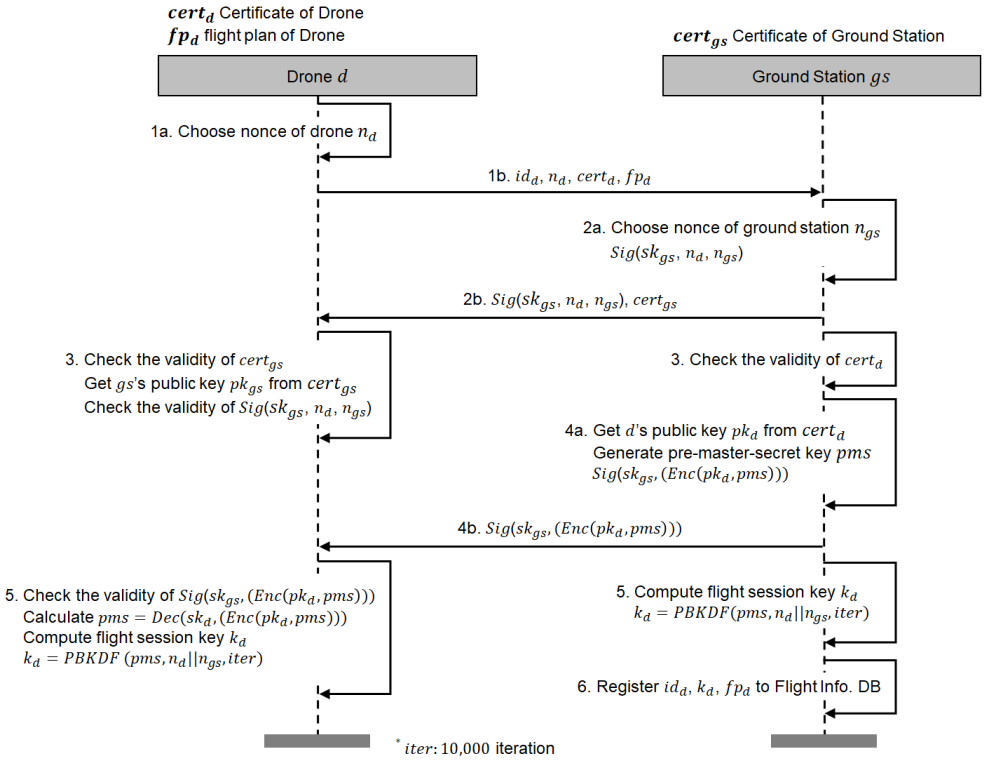


Figure 3. Mutual authentication and key agreement protocol.

5.2. Drone Authentication Protocol

To be authenticated from a ground station gs, the drone d just uses the flight session key k_d that is already registered to the flight information database that can be accessed by all ground stations. This scheme can be effective in reducing the communication overhead rather than using a certificate-based authentication scheme. In this authentication procedure, we consider two authentication scenarios: (1) the authentication procedure between a drone and a ground station, and (2) the authentication procedure between a drone and another drone. To authenticate the drone d, a ground station gs retrieves the registered flight session key k_d from the flight information database and then uses k_d to check the validity of authenticated code values computed by the drone d with the same key k_d. Figure 4 shows the first authentication procedure between d and gs.

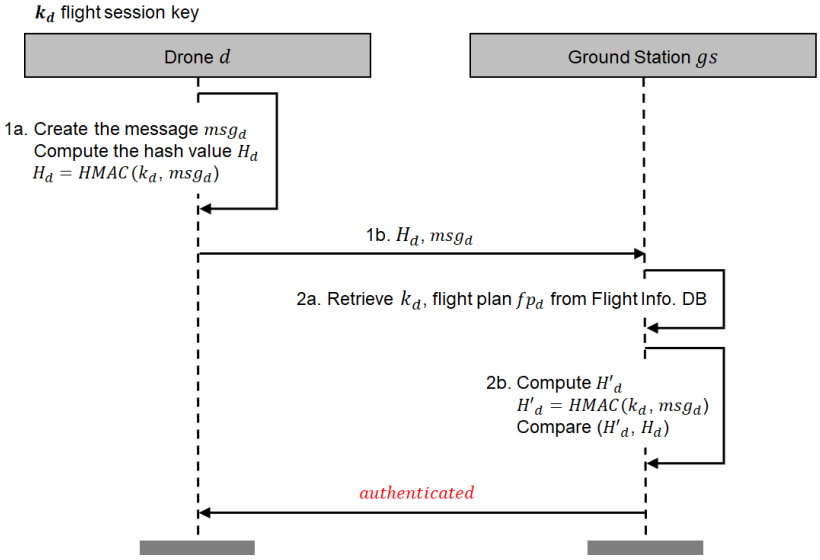


Figure 4. Authentication procedure between drone and ground station.

1. The drone d creates a message msg_d that contains the information about the drone d's (GPS) location and timestamp. The message also contains the drone d's identifier id_d as the message sender's identifier ('Source ID') and the ground station's identifier as the message recipient's identifier ('Destination ID') in order to prevent replay attacks and man-in-the-middle attacks. Then, d computes the authentication code H_d using a HMAC (hash-based message authentication code function) with k_d and msg_d and sends H_d with the original message msg_d to a ground station gs.
2. On receiving the message from the drone d, the ground station gs fetches the drone d's record with id_d from the flight information database and extracts the flight session key k_d from the record. For verifying the validity of the received H_d, gs computes $H'_d = HMAC(k_d, msg_d)$ using received message msg_d and the retrieved flight session key k_d. After computing H'_d, gs compares H'_d with the received authentication code H_d. If $H'_d = H_d$, gs believes that msg_d is not compromised and the drone d holds the flight session key k_d. After verifying the authentication code, gs makes a decision on whether or not to allow the drone d's flight entering its fly zone by checking whether this fly zone is contained in d's flight plan.

The message structure for the authentication is visually represented in Figure 5.

Figure 5. Message structure for authentication.

We summarize the description of each field in the message as follows.

- **Source ID.** This field represents the message sender's identity (i.e., the drone d's identity). The sender's identity can be used with the recipient's identifier to prevent man-in-the-middle attacks. In our prototype implementation, the size of this field is 4 bytes.
- **Destination ID.** This field represents the message recipient's identity. In the proposed protocol, the message recipient can be either a ground station or another drone. In our prototype implementation, the size of this field is 4 bytes.
- **Timestamp.** This field represents the time when the message was created. The timestamp can be used to ensure the freshness of messages. In our prototype implementation, the size of this field is 8 bytes.

- **GPS information.** This field represents the drone's current location information based on the GPS signal. This information can be used to trace the drone's location. In our prototype implementation, the size of this field is 24 bytes.
- **Authentication code.** This field represents the authentication code computed as $H_d = HMAC(k_d, msg_d)$. The authentication code is used to ensure message authenticity and integrity.

The proposed protocol can be extended to deal with the drone-to-drone authentication protocol. In the near future, drone-to-drone authentication would be necessary for some situations. For example, each drone should avoid collisions with other drones during flight. Therefore, it needs to interact and share its GPS information with other drones in real-time. From the perspective of cybersecurity, however, faked drone information can be intentionally generated and exchanged between drones [27]. To prevent such malicious activities, each drone should check the authenticity of other drones that communicate with. SENTINEL can also be a neat solution for drones to communicate with each other securely.

Figure 6 shows the authentication procedure between a drone d_1 and another drone d_2. Because d_1 and d_2 do not share a flight session key, the verifier drone d_2 checks the validity of the authentication code delivered from d_1 via a ground station gs.

1. The drone d_1 creates a message msg_{d_1} that contains the information about the drone d_1's (GPS) location and timestamp. The message also contains the drone d_1's identifier id_{d_1} as the message sender's identifier ('Source ID') and the other drone d_2's identifier as the message recipient's identifier ('Destination ID'). Then, d_1 computes the authentication code H_{d_1} using a HMAC (hash-based message authentication code function) with k_{d_1} and msg_{d_1} and sends H_{d_1} with the original message msg_{d_1} to the drone d_2.
2. On receiving the message from the drone d_1, the drone d_2 relays the message to a ground station gs.
3. On receiving the message from the drone d_2, the ground station gs fetches the drone d_1's record with id_{d_1} from the flight information database and extracts the flight session key k_{d_1} from the record. For verifying the validity of the received H_{d_1}, gs computes $H'_{d_1} = HMAC(k_{d_1}, msg_{d_1})$ using received message msg_{d_1} and the retrieved flight session key k_{d_1}. After computing H'_{d_1}, gs compares H'_{d_1} with the received authentication code H_{d_1}. If $H'_{d_1} = H_{d_1}$, gs believes that msg_{d_1} is not compromised and the drone d_1 holds the flight session key k_{d_1}. After verifying the authentication code, gs makes a decision on whether or not to allow the drone d_1's flight entering its fly zone by checking whether this fly zone is contained in d_1's flight plan. The ground station gs sends this result to the drone d_2.

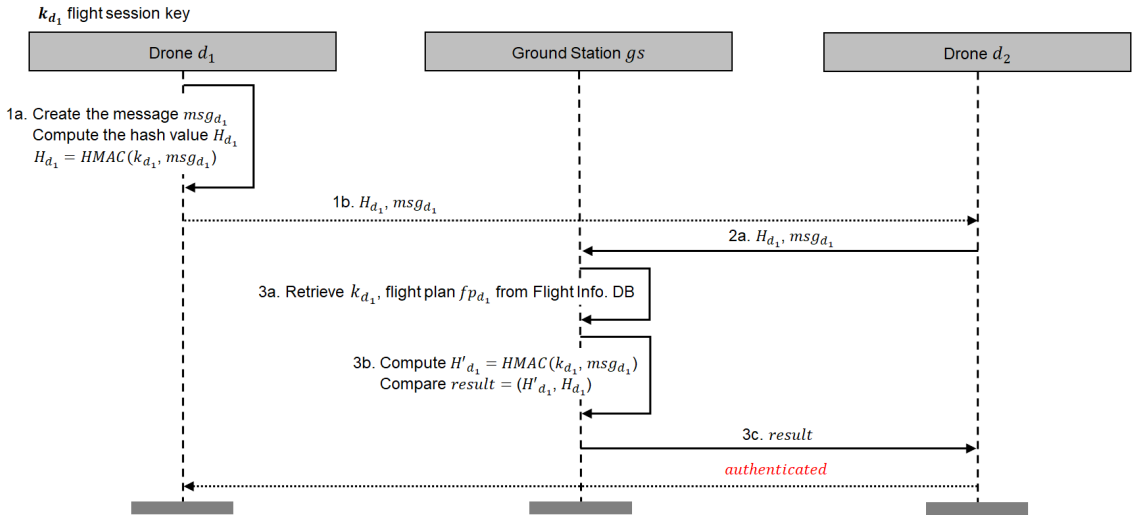


Figure 6. Authentication protocol between drone and drone.

6. Digital Certificate for SENTINEL

Public key infrastructure (PKI) is widely used to verify users and devices by using digital certificates on public communication channels such as the Internet. The X.509 v3 certificates [14] are most popularly used as the de facto standard to build PKI-based systems. However, the conventional X.509 v3 certificates may be too heavy in resource-constrained drones with respect to communication overhead, power consumption, and memory footprint because they are originally designed to accommodate comprehensive specifications of digital certificates with various options and extensibility. Thus we develop a lightweight certificate format that is viable for drones.

Figure 7 shows our lightweight certificate format for SENTINEL. Compared to the conventional X.509 v3 certificates, the proposed certificate only contains minimal information required to build the PKI in IoD environments. To further reduce the size of the certificate, we also use a binary format in the proposed certificate, instead of a human-readable text format adopted in the X.509 v3 certificates. The followings are the descriptions of each field in the proposed certificate.

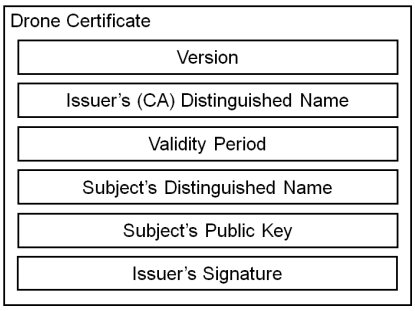


Figure 7. Structure of the proposed certificate format for drones.

- **Version.** This field represents a version of the certificate, which may be changed when the certificate is updated. We set the size of this field to 1 byte.
- **Issuer's (CA) Distinguished Name.** This field specifies the issuer's distinguished name. The size of this field is 2 bytes.
- **Validity Period.** This field specifies the period for which the certificate is valid. This certificate can no longer be used after being expired. We set the size of this field to 16 bytes.
- **Subject's Distinguished Name.** This field specifies the distinguished name of the subject, such as a drone's identifier and a ground station's identifier. The size of this field is 4 bytes.
- **Subject's Public key.** This field contains the subject's public key, and the length of the public key can be approximately 91 bytes. Unlike conventional X.509 v3 certificates, this field does not specify the type of a public key algorithm compatible with the public key because we assume a fixed algorithm. For drones, we recommend using ECDSA [28] to reduce their computational overhead.
- **Issuer's Signature.** This field contains the issuer's signature on this certificate, which will be used by recipients to verify the validity of this certificate. The size of this field is approximately 70 bytes.

In this paper, we define the size of each field for our prototype implementation. However, it can be flexibly adjusted depending on the requirements in PKI applications. Our experiment results in Section 7.2 show that the size of the proposed certificate was about 3.7 times smaller than the size of a typical X.509 v3 certificate.

7. Experiments

To show the feasibility and effectiveness of SENTINEL, we implemented a prototype and evaluated it through several experiments. As a baseline for comparison, we consider the TLS protocol with client authentication using X.509 v3 certificates, which is especially designed for IoT [29]. We

will refer to this protocol as "TLS for IoT." In both implementations of SENTINEL and "TLS for IoT", we use ECDSA with SHA-256 which is widely adopted by resource constrained devices in IoT [29].

7.1. Implementation

We used the following cryptographic algorithms and settings in our prototype implementation. For the digital signature algorithm, we used ECDSA with SHA-256 whose public key size is 256 bits in both implementations of SENTINEL and "TLS for IoT." To derive a flight session key, we used password-based key derivation function 2 (PBKDF2) [26] with HMAC-SHA256, where the number of iterations was set to 10,000, and the random salt was securely generated with cryptographically secure pseudorandom number generator (CSPRNG). To compute the authenticated code value of each drone's message with a flight session key, we used the HMAC-SHA256. We also used CSPRNG to generate drones' nonces (n_d), ground stations' nonces (n_{gs}), and the pre-master-secret key (pms).

For a prototype of SENTINEL, we implemented a drone program running on a MacBook Pro with Intel Core i7-7920HQ (3.10 GHz) running on macOS Catalina and used the 5G network interface to connect it to the Internet. We also implemented a ground station program running on a desktop with AMD Ryzen Threadripper 1950X (3.40 GHz) running on Windows 10 Pro and used Ethernet for network connectivity. For further information, the source code of our implementation is available at the public GitHub repository (https://github.com/rymuff/drone, see Supplementary Materials). The following sections present our evaluation results.

7.2. Evaluation—Lightweight Digital Certificate

To see how efficient our lightweight certificate is, we compared the size of our certificate with a conventional X.509 v3 certificate with necessary fields only and its compressed version using gzip. As shown in Table 2, the size of our certificate was reduced by 72.73% and 71.56% compared to the conventional X.509 v3 certificate and its gzip-compressed version, respectively, under the same cryptographic settings of ECDSA with SHA-256 and a 256-bits public key.

Table 2. Evaluation results of the size of certificates.

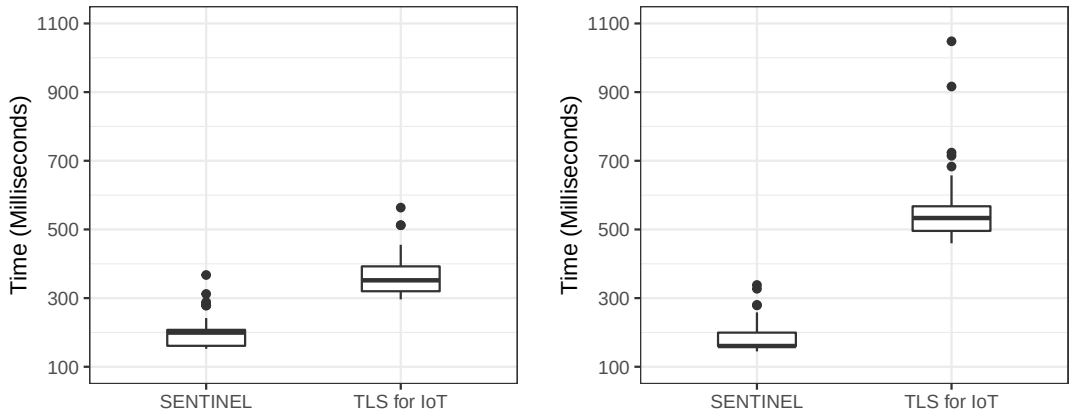
	SENTINEL	X.509	X.509 (gzip)
Size (bytes)	186	682	654

7.3. Evaluation—Mutual Authentication and Key Agreement Protocol

To evaluate the performance of the mutual authentication and key agreement protocol in SENTINEL, we measured the message size and the execution time of the entire procedure described in Figure 3. We also performed the same measurements on the "TLS for IoT" protocol with client authentication using an X.509 v3 certificate. For both systems, we conducted the same set of experiments 100 times and took the average. Table 3 summarizes the results. In addition, Figure 8a shows the distributions of the execution times for the mutual authentication and key agreement protocols, respectively, in SENTINEL and "TLS for IoT."

Table 3. Evaluation results of mutual authentication and key agreement protocol.

		SENTINEL	TLS for IoT
Time (ms)	**Avg.**	198.06	359.66
	Std.	39.25	50.80
Message size (bytes)	**Send**	426	708
	Receive	192	825
	Total	618	1533



(a) Mutual authentication and key agreement (b) Drone authentication

Figure 8. Execution time comparison between SENTINEL and "TLS for IoT."

For SENTINEL, the average execution time was 198.06 milliseconds with the standard deviations of 39.25 milliseconds, and the total number of bytes sent and received was 618 bytes. For the "TLS for IoT" protocol with client authentication, on the other hand, the average execution time was 359.66 milliseconds with the standard deviations of 50.80 milliseconds, and the total number of bytes sent and received was 1,533 bytes. In comparison, the execution time of SENTINEL was 161.60 milliseconds faster than the "TLS for IoT" protocol on average. Furthermore, the total number of bytes sent and received in SENTINEL was 915 bytes smaller than the "TLS for IoT" protocol.

7.4. Evaluation—Drone Authentication Protocol

We also evaluated the performance of the authentication protocol in SENTINEL and compared it with the "TLS for IoT" protocol with client authentication. We assumed that the size of the message (msg_d) to be authenticated was 40 bytes. In the case of the "TLS for IoT" protocol with client authentication, the handshaking procedure was first performed between the drone d and the ground station gs, and the drone d then creates a message for authentication by signing the message with its private key sk_d, while in SENTINEL, the drone d just computes the message authentication code using the flight session key without the handshaking procedure. Under this assumption, we performed measurements on the authentication protocol of both SENTINEL (described in Figure 4) and the conventional system. As shown in Table 4, the average execution time of SENTINEL was 178.57 milliseconds with the standard deviation of 35.07 milliseconds, and the message sizes were 72 bytes in total. For the "TLS for IoT" protocol, on the other hand, the average execution time was 549.75 milliseconds, with the standard deviation of 84.27 milliseconds, and the message size was 1644 bytes in total. In comparison, the execution time of SENTINEL (178.57 milliseconds) was about 3.1 times faster than the "TLS for IoT" protocol (549.75 milliseconds) on average. Furthermore, the total size of the messages used in SENTINEL was 1572 bytes smaller than the "TLS for IoT" protocol. In addition, Figure 8b shows the distributions of the execution times for the drone authentication protocols, respectively, in SENTINEL and "TLS for IoT."

Table 4. Evaluation results of drone authentication protocol.

		SENTINEL	TLS for IoT
Time (ms)	**Avg.**	178.57	549.75
	Std.	35.07	84.27
Message size (bytes)	**Total**	72	1644

8. Security Analysis

8.1. Formal Security Analysis Using ProVerif

To analyze the security of SENTINEL, we present a brief formal security evaluation of the SENTINEL using ProVerif [30], which is an automatic symbolic protocol verifier, and popularly used for verifying cryptographic protocols (e.g., [31,32]). First of all, we define a set of cryptographic primitives F that is used to verify the key agreement procedure. F is used for the drone as well as the ground station (see the ProVerif code in Listing 1). There are two types of encryption methods: `ENC` for symmetric key encryption and `AENC` for public key encryption. `PBKDF` is used to create the flight session key, and `salt` consists of the nonce of the drone and the ground station.

Listing 1. List of functions required for the key agreement procedure for both drone and ground station.

```
Functions:
F={SIGN, CheckSIGN, SIGN_KEY, SIGN_CERT, GetCERT, CheckSIGN_CERT,
ENC, DEC, AENC, ADEC, PBKDF}
SIGN(private key, data) = signed value.
CheckSIGN(signed value, public key) = True or False.
SIGN_KEY(key, private key) = singed value.
SIGN_CERT(certificate, private key) = singed value.
GetCERT(signed value) = certificate.
CheckSIGN_CERT(signed value, public key) = True or False.
ENC(key, data) = cipher text.
DEC(key, cipher text) = data.
AENC(public key, data) = cipher text.
ADEC(private key, cipher text) = data.
PBKDF(secret value, salt, iteration) = hashed value.
```

Listings 2 and 3 show the procedures performed for key agreement between drone d and ground station gs. In our ProVerif model, d initially holds its private key sk_d, public key pk_d, certificate $cert_d$ and CA's public key pk_{CA}. gs also holds its private key sk_{gs}, public key pk_{gs}, certificate $cert_{gs}$ and CA's public key pk_{CA}. Both $cert_d$ and $cert_{gs}$ can be verified using pk_{CA}.

As mentioned in Section 3.2, we set an active attacker who has complete control of the communication channel between d and gs. The attacker can read, modify, delete and inject messages. The attacker is also able to manipulate the data. In the step 2a of Figure 3, gs sends its random nonce n_{gs} along with the drone's random nonce n_d after being signed with the private key of ground station sk_{gs}. We defined the sign function (denoted as `SIGN(private key, data) = signed value`) and used it as `Sig(`sk_{gs}`, signed nonce)` in Listing 3. To check the validity of the signed value, d uses both `GetCERT` and `GetPKEY`. In the step 3 of Figure 3, d first obtains gs's public key pk_{gs} using `GetCERT(received` $cert_{gs}$`)` in Listing 2. Then, d uses `CheckSIGN(singed nonce,` pk_{gs}`)` to check the validity of the singed nonce. To create a flight session key k_d, both d and gs use `PBKDF(`pms`, concatenated nonce (`$n_d \| n_{gs}$`))`.

Listing 2. Procedure performed on drone d for key agreement.

```
Stored information:
(sk_d, pk_d, cert_d, pk_CA)
Process: Drone≙
new id.
new n_d.
out(ch, (id, n_d, cert_d)).
in(ch, (signed nonce, received cert_gs)).
let cert_gs = GetCERT(received cert_gs) in
let valid cert_gs = CheckSIGN_CERT(received cert_gs, pk_CA) in
if cert_gs = valid cert_gs then
let pk_gs = GetPKEY(cert_gs) in
let received nonce = GetMESS(signed nonce) in
let valid received nonce = CheckSIGN(signed nonce, pk_gs) in
if received nonce = valid received nonce then
in(ch, signed pms).
let encrypted pms = GetMESS(signed pms) in
let valid encrypted pms = CheckSIGN(signed pms, pk_gs) in
if encrypted pms = valid encrypted pms then
let pms_d = ADEC(encrypted pms, sk_d) in
let k_d = PBKDF(pms_d, received nonce) in
event acceptsDrone(k_d).
out(ch, ENC(s, k_d)).
```

Listing 3. Procedure performed on ground station gs for key agreement.

```
Stored information:
(sk_gs, pk_gs, cert_gs, pk_CA)
Process: Ground station≙
in(ch, (id, n_d, received cert_d)).
new n_gs.
let concatenated nonce = Conc(received nonce, n_gs) in
let signed nonce = SIGN(concatenated nonce, sk_gs) in
out(ch, (signed nonce, cert_gs).
let cert_d = GetCERT(received cert_d) in
let valid cert_d = CheckSIGN_CERT(received cert_d, pk_CA) in
if cert_d = valid cert_d then
let pk_d = GetPKEY(cert_d) in
new pms_gs.
let k_gs = PBKDF(pms_gs, concatenated nonce) in
let signed pms_gs = SIGN(AENC(pms_gs, pk_d), sk_gs) in
out(ch, signed\ pms_gs).
in(ch, received secret).
let decrypted secret = DEC(received secret, k_gs) in
event termGS(k_gs).
```

Listing 4 shows the queries and verification results in ProVerif. To confirm the flight session key correctly shared, we used two queries. The first query result of `RESULT not attacker(s[]) is true` means that the attacker failed to obtain secret `s[]`, leading to the secrecy of the flight session key k_d. This is because the attacker can finally obtain secret `s` if the attacker obtains k_d.

Listing 4. Attack trace for obtaining the shared key.

```
Query:
query not attacker(s[])
query x: key; inj-event(termGS(x))==>inj-event(acceptsDrone(x)).
Result:
RESULT not attacker(s[]) is true.
RESULT inj-event(termGS(x_79))
==>inj-event(acceptsDrone(x_79)) is true.
```

To check that *d* and *gs* share the same flight session key correctly, we defined `query x: key; inj-event(termGS(x))==>inj-event(acceptsDrone(x))` to check whether the key is same or not. The query result of `RESULT inj-event(termGS(x_79))==>inj-event(acceptsDrone(x_79)) is true.` indicates that *d* and *gs* successfully share the same flight session key for authentication.

8.2. Informal Security Analysis

We also provide a security analysis for SENTINEL against various security attacks. Here we briefly summarize the security properties that are provided by SENTINEL compared with existing authentication protocols for IoT or drones based on the threat model (see Section 3.2). Table 5 summarizes the results, demonstrating that SENTINEL provides all the key features for the IoD environment.

Table 5. Summary and comparison of existing authentication protocols for IoD and our solution ('-' indicates that the paper did not mentioned them and 'N/A' indicates that the attack cannot be performed because there is no session key).

Security Features	Srinivas et al. [5]	Zhang et al. [6]	Wazid et al. [4]	Won et al. [23]	Tian et al. [19]	Chen et al. [24]	SENTINEL
Mutual authentication	✓	✓	✓	✓	✗	✓	✓
Pseudonymity/Anonymity	✓	✓	✓	✗	✗	✓	✓
Revocability	✓	✓	✓	✗	✗	✗	✓
Session key agreement	✓	✓	✓	✓	✗	✓	✓
Man-in-the middle attack	✓	✓	✓	✓	-	✓	✓
Replay attack	✓	✓	✓	-	✓	✓	✓
Impersonation attack	✓	✓	✓	-	-	✓	✓
Known session key attack	-	✓	-	-	N/A	-	✓

In the following, we clarify how SENTINEL provides each of the security features mentioned in Table 5.

- **Mutual authentication.** Both the drone and the ground station have certificates issued by CA. During the key agreement process, both entities exchange their certificates and check the validity of the certificates. Consequently, the drone and the ground station authenticate each other if their certificates are valid.
- **Pseudonymity.** During the authentication process, the drone sends a message that contains its identity in SENTINEL. For preserving the privacy of the drone, a pseudonym can be used instead of the real identity of the drone. In this way, the ground station can still check if the drone has been authorized for its flight without exposing the drone's real identity.
- **Revocability.** As mentioned in Section 5, both the drone and the ground station can check whether the opposing party's certificate is revoked using a certificate validation service such as an online certificate status protocol (OCSP).
- **Session key agreement.** The drone and the ground station perform the key agreement process to make a flight session key (see Figure 3).

- **Man-in-the middle attack.** Under the mutual authentication, SENTINEL guarantees to resist against man-in-the-middle attacks. Both the drone and ground station validate each other using the certificate.
- **Replay attack.** The timestamp is contained in the message for authentication that represents the time to create the message. With the timestamp, the message cannot be reused because the ground station can check the freshness of the message.
- **Impersonation attack.** A malicious drone can try to impersonate a legitimate drone d using id_d and $cert_d$. Although the malicious drone uses id_d and $cert_d$, it is not possible to obtain the pre-master-secret key without the drone d's private key.
- **Known session key attack.** Flight session keys are always newly created with random parameters such as n_d, n_s, and pms. Therefore, it is not possible to succeed with a known session key attack.

9. Conclusions

We proposed SENTINEL to verify the authenticity of drones for detecting unauthorized drones in fly zones of the IoD environment. In IoD, it is crucial to authenticate drones in real-time due to drones' fast movements, and lightweight protocols are needed for supporting resource-constrained drones. SENTINEL provides secure and efficient protocols with a lightweight certificate format to reduce traffic and computational overheads to meet such requirements in the IoD infrastructure.

To demonstrate the feasibility of the proposed framework, we implemented a prototype of SENTINEL and compared it with the conventional "TLS for IoT" protocol with client authentication. The experiment results demonstrate that SENTINEL could bring significant benefits concerning performance. The execution time of the authentication protocol in SENTINEL is about 3.1 times faster than the "TLS for IoT" protocol on average. We also formally verified the security of SENTINEL using ProVerif, a well-known protocol verification tool.

Supplementary Materials: The source codes of our implementation are available in the public GitHub repository (https://github.com/rymuff/drone). The GitHub repository also contains the ProVerif code used for security analysis of SENTINEL.

Author Contributions: Methodology, G.C. and J.C.; Software, J.C.; Formal analysis, G.C.; Writing—original draft, G.C. and J.C.; Writing—review & editing, S.H. and H.K.; Supervision, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD190016ED).

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

Abbreviations

The following abbreviations are used in this manuscript:

ASN.1	Abstract syntax notation one
CA	Certificate authority
CSPRNG	Cryptographically secure pseudorandom number generator
DER	Distinguished encoding rules
ECDSA	Elliptic curve digital signature algorithm
GS	Ground station
HMAC	Hash-based message authentication code
IoD	Internet of drones
IoT	Internet of things
LTE	Long-term evolution
MAC	Message authentication code

OCF	Open connectivity foundation
OCSP	Online certificate status protocol
PBKDF	Password-based key derivation function
PIN	Personal identification number
PKI	Public key infrastructure
SHA	Secure hash algorithm
TLS	Transport layer security
UAV	Unmanned aerial vehicle

References

1. Symington, S. Amazon.com Has Officially Begun Drone Delivery. Available online: https://www.fool.com/investing/2016/12/20/amazoncom-has-officially-begun-drone-delivery.aspx (accessed on 18 March 2020).
2. Gang, J. Drone Use in the Entertainment Industry and Beyond. Available online: https://thebottomline.as.ucsb.edu/2018/09/drone-use-in-the-entertainment-industry-and-beyond (accessed on 18 March 2020).
3. Shear, M.D.; Schmidt, M.S. White House Drone Crash Described as a U.S. Worker's Drunken Lark. Available online: https://www.nytimes.com/2015/01/28/us/white-house-drone.html (accessed on 18 March 2020).
4. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V.; Rodrigues, J.J. Design and analysis of secure lightweight remote user authentication and key agreement scheme in Internet of drones deployment. *IEEE Internet Things J.* **2018**, *6*, 3572–3584. [CrossRef]
5. Srinivas, J.; Das, A.K.; Kumar, N.; Rodrigues, J.J. TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6903–6916. [CrossRef]
6. Zhang, Y.; He, D.; Li, L.; Chen, B. A lightweight authentication and key agreement scheme for internet of drones. *Comput. Commun.* **2020**, *154*, 455–464. [CrossRef]
7. Lloyd, B.; Simpson, W. PPP Authentication Protocols. RFC 1334. 1992. Available online: https://tools.ietf.org/html/rfc1334/ (accessed on 27 March 2020).
8. Simpson, W. PPP Challenge Handshake Authentication Protocol (CHAP). RFC 1994. 1996. Available online: https://tools.ietf.org/html/rfc1994/ (accessed on 27 March 2020).
9. Aboba, B.; Blunk, L.; Vollbrecht, J.; Carlson, J.; Levkowetz, H.E. Extensible Authentication Protocol (EAP). RFC 3748. 2004. Available online: https://www.hjp.at/doc/rfc/rfc3748.html (accessed on 27 March 2020).
10. The OCF Security Specification. 2020. Available online: https://openconnectivity.org/developer/specifications/ (accessed on 27 March 2020).
11. IoTivity Wiki. Available online: https://iotivity.org/ (accessed on 27 March 2020).
12. Sciancalepore, S.; Capossele, A.; Piro, G.; Boggia, G.; Bianchi, G. Key Management Protocol with Implicit Certificates for IoT Systems. In Proceedings of the Workshop on IoT Challenges in Mobile and Industrial Systems, Florence, Italy, 18 May 2015.
13. Porambage, P.; Schmitt, C.; Kumar, P.; Gurtov, A.; Ylianttila, M. Two-phase authentication protocol for wireless sensor networks in distributed IoT applications. In Proceedings of the IEEE Conference on Wireless Communications and Networking, Istanbul, Turkey, 6–9 April 2014.
14. Housley, R.; Ford, W.; Polk, W.; Solo, D. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459. 1999. Available online: https://rfc-editor.org/rfc/rfc2459.txt (accessed on 27 March 2020).
15. Kwon, H.; Raza, S.; Ko, J. POSTER: On Compressing PKI Certificates for Resource Limited Internet of Things Devices. In Proceedings of the 2018 on Asia Conference on Computer and Communications Security, Incheon, Korea, 4–8 June 2018.
16. Gharibi, M.; Boutaba, R.; Waslander, S.L. Internet of Drones. *IEEE Access* **2016**, *4*, 1148–1162. [CrossRef]
17. Lin, C.; He, D.; Kumar, N.; Choo, K.K.R.; Vinel, A.; Huang, X. Security and Privacy for the Internet of Drones: Challenges and Solutions. *IEEE Commun. Mag.* **2018**, *56*, 64–69. [CrossRef]
18. Ni, J.; Lin, X.; Zhang, K.; Shen, X. Privacy-preserving real-time navigation system using vehicular crowdsourcing. In Proceedings of the 2016 IEEE 84th Vehicular Technology Conference (VTC-Fall), Montréal, QC, Canada, 18–21 September 2016; pp. 1–5.
19. Tian, Y.; Yuan, J.; Song, H. Efficient privacy-preserving authentication framework for edge-assisted Internet of Drones. *J. Inf. Secur. Appl.* **2019**, *48*, 102354. [CrossRef]

20. Greenpeace Crashes Superman-Shaped Drone into French Nuclear Plant. 2018. Available online: https://www.reuters.com/article/us-france-nuclear-greenpeace/greenpeace-crashes-superman-shaped-drone-into-french-nuclear-plant-idUSKBN1JT1JM (accessed on 28 March 2020).
21. Gatwick Airport Drone Incident. 2018. Available online: https://en.wikipedia.org/wiki/Gatwick_Airport_drone_incident (accessed on 28 March 2020).
22. Drone Filmed People's Pin Codes at Co Antrim ATM. 2016. Available online: https://www.belfasttelegraph.co.uk/news/northern-ireland/drone-filmed-peoples-pin-codes-at-co-antrim-atm-34945847.html (accessed on 28 March 2020).
23. Won, J.; Seo, S.; Bertino, E. Certificateless cryptographic protocols for efficient drone-based smart city applications. *IEEE Access* **2017**, *5*, 3721–3749. [CrossRef]
24. Chen, C.L.; Deng, Y.Y.; Weng, W.; Chen, C.H.; Chiu, Y.J.; Wu, C.M. A traceable and privacy-preserving authentication for UAV communication control system. *Electronics* **2020**, *9*, 62. [CrossRef]
25. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
26. Kaliski, B. *PKCS# 5: Password-Based Cryptography Specification Version 2.0*; RFC 2898; RSA Laboratories: Bedford, MA, USA, 2000. Available online: https://tools.ietf.org/html/rfc2898/ (accessed on 30 June 2019).
27. Berges, P.M. Exploring the Vulnerabilities of Traffic Collision Avoidance Systems (TCAS) Through Software Defined Radio (SDR) Exploitation. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2019.
28. Johnson, D.; Menezes, A.; Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Secur.* **2001**, *1*, 36–63. [CrossRef]
29. Suárez-Albela, M.; Fernández-Caramés, T.M.; Fraga-Lamas, P.; Castedo, L. A Practical Performance Comparison of ECC and RSA for Resource-Constrained IoT Devices. In Proceedings of the Global Internet of Things Summit, Bilbao, Spain, 4–7 June 2018.
30. Blanchet, B. Automatic verification of correspondences for security protocols. *J. Comput. Secur.* **2009**, *17*, 363–434. [CrossRef]
31. Bhargavan, K.; Blanchet, B.; Kobeissi, N. Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate. In Proceedings of the IEEE Symposium on Security and Privacy, San Jose, CA, USA, 22–26 May 2017.
32. Kobeissi, N.; Bhargavan, K.; Blanchet, B. Automated Verification for Secure Messaging Protocols and their Implementations: A Symbolic and Computational Approach. In Proceedings of the IEEE European Symposium on Security and Privacy, Paris, France, 26–28 April 2017.