Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6653883, 17 pages
https://doi.org/10.1155/2021/6653883

Research Article

A Random Label and Lightweight Hash-Based Security Authentication Mechanism for a UAV Swarm

Feng Hu, Hongyan Qian, and Liangjun Liu

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

Correspondence should be addressed to Hongyan Qian; qhy98@nuaa.edu.cn

Received 20 November 2020; Accepted 26 May 2021; Published 1 July 2021

Academic Editor: Junwu Zhu

Copyright © 2021 Feng Hu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nowadays, the application of a UAV swarm is becoming more and more widespread in the military field, and more and more attention is paid to the security of mission resource allocation. However, the relay node forwarding in the wireless transmission process brings greater risks to data leakage, and the computing power and energy of the UAV consumption is limited, so a lighter solution is required. This paper proposes a mechanism for the safe allocation of UAV swarm mission resources based on random labels. Each task has a random label to solve the problem of database security and wireless transmission security in the process of UAV task assignment. Furthermore, a lightweight stream cipher encryption scheme is illustrated to ensure the security of the UAV database. The irreversible hash function SHA-256 and the lightweight foam structure hash function SPONGENT-128 are used to generate random labels and then allocate task resources. In the case of energy consumption, it reduces the possibility of the enemy successfully obtaining private data. The simulation results show that the scheme has good performance in terms of security and has better performance than existing methods in terms of throughput and delay, without increasing too much energy consumption.

1. Introduction

UAV networks are currently attracting more and more attention. The coordination and collaboration between multiple UAVs and ground base stations have established UAV swarm systems, which are easy to deploy, low in purchase and maintenance costs, and large in coverage and battery capacity. Small, therefore limited flight time but high maneuverability, so it is a practical choice for civilian and military applications, especially in the military field [1], can be used for environmental and natural disaster monitoring, border patrols, emergency assistance, search and rescue missions, and cargo delivery. At the same time, the UAV swarm can also be deployed as an aerial base station. A leading UAV serves as a mobile control station for command and information gathering tasks. The surrounding member UAVs execute commands. Such a team has the ability to execute ordinary equipment, but cannot complete the task [2]. As the number of UAVs increases, UAV networks can provide collaboration strategies, but there are still challenges in airborne networks in terms of information sharing and information security in airborne networks. Fast-moving nodes will produce constantly changing topologies in the network, leading to frequent disconnections and interruptions in the network; in an airborne network, each node is regarded as an autonomous embedded system with local- and network-level resource constraints. The system, as the autonomous system develops through interaction with the environment, also becomes vulnerable to malicious control by competitors [3]. The application of the UAV swarm in the military field is more and more popular as the demands grow more and more. In particular in terms of missions on collecting sensitive information or confidential tasks, the security and integrity of mission resource assignment on UAV during a flight are still a problem needed to be resolved [4, 5]. Not only does the content of the mission resources need to be protected but also the transmission data.

Therefore, a security strategy is bound to be needed, which can protect the integrity of its embedded system and allow it to be verified to make the system trustworthy. Although there are many researches on the safe communication and information sharing of UAVs, most of them do not take into account the effective limited computing power of UAVs, or the high latency cannot obtain better efficiency in the field of military applications. The flying mobile ad hoc network is a complex system. There is much room for improvement from the physical and network layers. Therefore, it is necessary to design and implement a large-scale UAV swarm security certification model to effectively protect the database security and wireless transmission security during the mission of the UAV. In this paper, we attach labels to every certain mission resource, and we propose the stream cipher-based database security approach to protect the content of the mission in the beginning and the two-hash-function-based security model for the process of transmission to prevent the system from intercepting or eavesdropping. The UAV swarm can be divided into several task units during the execution of tasks; each unit contains the following:

(i) A leader UAV plays the role of commanding, collecting, and distributing messages. It is equivalent to a mobile ground station, with stronger computing capabilities, and also acts as the core of the entire team

(ii) Several surrounding member UAVs as the relay nodes are responsible for receiving the leader's messages and forwarding data packets, keep in touch with the leader, and are verified by the leader every certain period

(iii) The peripheral UAV followers around the relay nodes are responsible for receiving messages from the relay nodes and are verified by the relay nodes in a higher frequency

In the process of performing the mission, the swarm will inevitably encounter problems such as accidental damage or being attacked by malicious UAVs during the flight and eavesdropping. Therefore, the safety of the leader of the UAV is imperative. If the leader is compromised, then the entire system will fall short. Therefore, in order to protect the security of the leader, we adopt the RC4 lightweight encryption scheme to make sure that the leader's database is no longer exposed to the enemy's line of sight, and the followers around the leader cannot be trusted fully; in order to make their status always clearly informed by the system, we use two hash functions to protect the transmission data privacy, and label verification is done every certain period to ensure the reliability of the follower's identity. This paper takes the safety and energy consumption limits of UAV swarms into account and uses SHA-256 and lightweight hash function—SPONGENT-128—to protect the security of the system to the greatest extent, while reducing the probability of malicious attacks and information acquisition.

We have done a lot of research and found that most of the existing UAV security solutions cannot take the balance between task efficiency and energy consumption into consideration. Some related experts and scholars have proposed some encryption protocols, such as LCAP protocol [6] and hash lock [7]. However, these protocols have hidden security risks of being attacked in the network. For these problems, the idea of a random label was utilized to build a security authentication model for a large-scale UAV swarm in our previous work [8], where a security authentication mechanism based on a random label was proposed, and the processing logic of data packets in the security authentication process of the UAV swarm was explained. Different from [8], we propose and implement a UAV swarm security authentication mechanism based on the random label and lightweight hash in this paper, so that the information sent by the leader can be encrypted by the RC4 encryption scheme before being distributed, which protects the database of the system. After the member UAVs receive the information, they will decrypt it firstly and then generate random labels corresponding to SHA-256 (hash-1) or SPONGENT-128 (hash-2) to verify their identity. Since the whole system contains mostly UAVs with calculation limitation, we try to adapt lightweight hashing schemes. And the verification between the systems is completed by irreversible hashing correspondingly to generate random labels that are difficult to decipher. Moreover, in order to prevent the enemy from using the same encryption scheme to detect the decryption method, the leader changes the custom key every other period of time. In this way, even if the enemy intercepts this information, there is no way to know the decryption scheme, which greatly reduces the probability of the UAV swarm being infiltrated without increasing the consumption of a lot of energy. Experimental results show that the model enables the entire UAV system to achieve better performance in terms of ensuring robustness and task completion efficiency.

The main contents of our work are as follows:

(i) This paper introduces the random labels, and uses a certain label to generate and verify uncertain label to ensure the security of UAV swarm system

(ii) A security model of large-scale UAV swarms based on random labels is proposed. A lightweight encryption scheme is used to protect the security of the database. For UAVs with complex computing capabilities, an improved hash function is used for label generation and verification. For UAVs with weak computing power, a lightweight hash function generation and verification is proposed to protect the safety of data transmission

(iii) The safety analysis of the proposed scheme is carried out, and the analysis results show that the model proposed in this paper has good safety performance

(iv) A simulation experiment was carried out on the proposed safety model, and its performance was compared with the existing mechanism extensively. The simulation results show that the proposed scheme can achieve better performance in terms of task efficiency and privacy protection

The rest of the paper is organized as follows. The related works are illustrated in Section 2. The preliminaries and

system model are presented in Section 3. Section 4 describes the design details of our proposed secure mechanism for mission resource allocation and the safety analysis of the system. Simulation results and analysis are illustrated in Section 5. Finally, important conclusion is drawn in Section 6.

2. Related Works

The main security threats faced by UAVs include [9] wireless signal hijacking and jamming, sensor network attacks, and GPS spoofing [5].

Since the communication of the UAV is open, it is easy for the enemy to attack and interfere with the wireless signal, and this can directly affect the normal operation of the UAV. For this kind of attack, [5] proposed the eCLSC-TKEM communication protocol based on the key encapsulation mechanism; [10] constructed a UAV monitoring system to determine whether the UAV has been infiltrated and is not under someone else's control; [11] proposed two private communication methods that can be used in a mobile ad hoc network for UAV flight. The security of the wireless sensor network composed of a UAV swarm also has great vulnerabilities, such as database security and the security of the wireless transmission process, which are easily intercepted, cracked, analyzed, and attacked by the enemy. There are some solutions for UAV communication security in the existing literature, which are mainly implemented through physical layer security, secure routing, and security algorithms.

Physical layer security has studied several technologies to realize the positive privacy rate of the UAV communication system, such as artificial noise [12], power control, k-anonymity algorithm for UAV location security [13], flexible location privacy protection [5], and the construction of protected areas [14]. Secure routing enables FANET to transmit information in a more secure routing, such as that based on a geographic location algorithm [15, 16], based on a multipath planning algorithm [17, 18], based on a swarm secure routing protocol algorithm [19], based on a security cryptographic algorithm [18, 20], and based on an intelligent algorithm [21]. The security algorithm encrypts data and authenticates the identity of the UAV to achieve more security of FANET. For example, the ChaCha20 encryption algorithm and Poly1305 for encryption authentication have become popular alternative methods in the industry to perform advanced encryption (AE) [22].

In addition, the attacker can also forge GPS signals to deceive the UAV's GPS, causing the UAV's navigation system or position coordinates to point to an offset position [23, 24], causing the UAV to fly to the wrong target location, affecting the efficiency of task completion. However, most of the existing work focuses on UAV-to-ground systems in a two-dimensional space, and there is only one communication model from the ground to the air. The security of the UAV-UAV (A2A) system in a three-dimensional (3D) space will be more complicated, because the receiver or eavesdropper works in all directions. At present, the three-dimensional security model has not been well studied and understood, leaving an open question. At the same time, because of the limited computing power of UAVs, most of the existing security algorithms have high latency and high computational complexity. This paper requires a lighter and faster security strategy to ensure the safety of the UAV swarm system.

3. Preliminaries and System Models

3.1. System Model. After taking off from the ground station, the UAV swarm may be disconnected. At this time, the swarm is divided into several small mission units, and each mission unit has a leader equipped with higher ability to carry out complicated calculation and analysis. The leader collects the information sent by the following UAVs, distributes new packets, and acts as the mobile ground station. As shown in Figure 1, a circle of UAVs around the leader is relay nodes (leader surrounding); they are responsible for collecting the information sent by the leader and forwarding the data packet to the peripheral follower. The relay nodes directly communicate with the leader and directly undergo the verification by the leader, but the relay node needs to verify the identity of the more peripheral members. Therefore, we allow the leader UAV to have higher computing power. They are more capable of processing more complex computing logic than the member UAVs that follow. But the computing power of the relay nodes and peripheral UAV is limited, and in order to save energy consumption, we require a lighter solution to support the more frequent authentication process [25, 26].

First of all, in order to prevent the leader from attracting firepower due to system centralization and to ensure the security and credibility of the leader's database, the certain label issued by the leader needs to be simply symmetrically encrypted through RC4 to play the role of query protection, and the key of the symmetrically encrypted plan usually only need to be XORed once to get the original text without causing excessive energy consumption; secondly, in order to ensure data security during the wireless transmission of the entire system, similar to the work [27, 28], in addition to the certain label (clabel) used to track the conditions of the information packet, under this model, we introduce an uncertain label (ulabel) to allow the UAV to prove its identity [26].

When the relay node receives the data packet sent by the leader, it needs to decrypt and get the certain label first and then use the improved SHA-256 scheme to generate the corresponding uncertain label (ulabel) and send it back to the leader every certain period to verify the identity. At the same time, the data packet is forwarded to the next hop, which is the peripheral follower UAV. After the peripheral UAV receives the data packet, it decrypts the clabels first, then uses the lightweight SPONGENT-128 scheme to generate an uncertain label (ulabel), and then transmits it back to the relay nodes for identification. In the verification process, since the hash is irreversible, the verifier needs to use the same scheme to generate a random label for result comparison. If they are consistent, the verification is passed; otherwise, the system triggers an alarm.

When the UAV swarm takes off from the ground control station, it disconnects from the ground station. At this time, the leading UAV will assign clabel to each member UAV for real-time tracking and identity verification. Because the

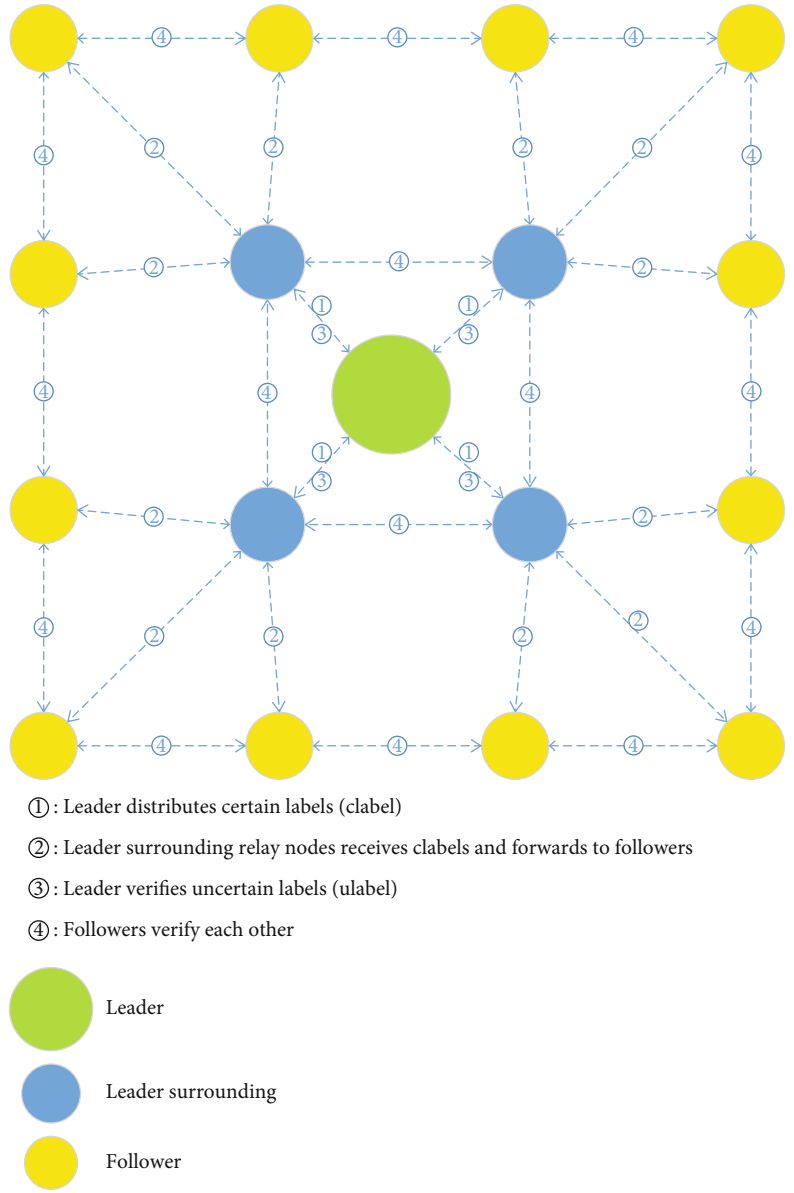


Figure 1: System model.

safety of the leading UAV is of utmost importance, in order to ensure the safety of the clabel issued, the leading UAV first processes the clabel with a lightweight stream cipher encryption scheme and then forwards it. In this way, even if the enemy breaks through the leading UAV, they cannot know the accurate clabel, which reduces the probability of the enemy attacking the leading UAV to a certain extent. Meanwhile, the lightweight encryption scheme will not bring a lot of energy to the system consumption.

3.2. Stream Cipher. A stream cipher is a common encryption algorithm based on XOR operation, which only operates one byte at a time. Stream cipher encryption is a widely used algorithm in lightweight encryption algorithms. It has fast speed and is convenient for hardware to realize along with the advantages of less memory and less error propagation. One of the most representative algorithms is the Rivest Cipher (RC4) algorithm [29]. Taking into account the energy consumption of UAVs, we use this lightweight encryption algorithm to protect the safety of the clabel, thereby protecting the safety of the leading UAV.

3.3. SHA-256. SHA (Secure Hash Algorithm) is a series of cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST), including SHA-1, SHA-224, and SHA—variants such as SHA-256, SHA-384, and SHA-512. It is mainly applicable to the Digital Signature Algorithm (DSA) defined in the Digital Signature Standard (DSS). Because the security of SHA-1 has been seriously questioned because it has been cracked, but the security of the SHA-2 series can still resist most attacks, considering the impact of the length

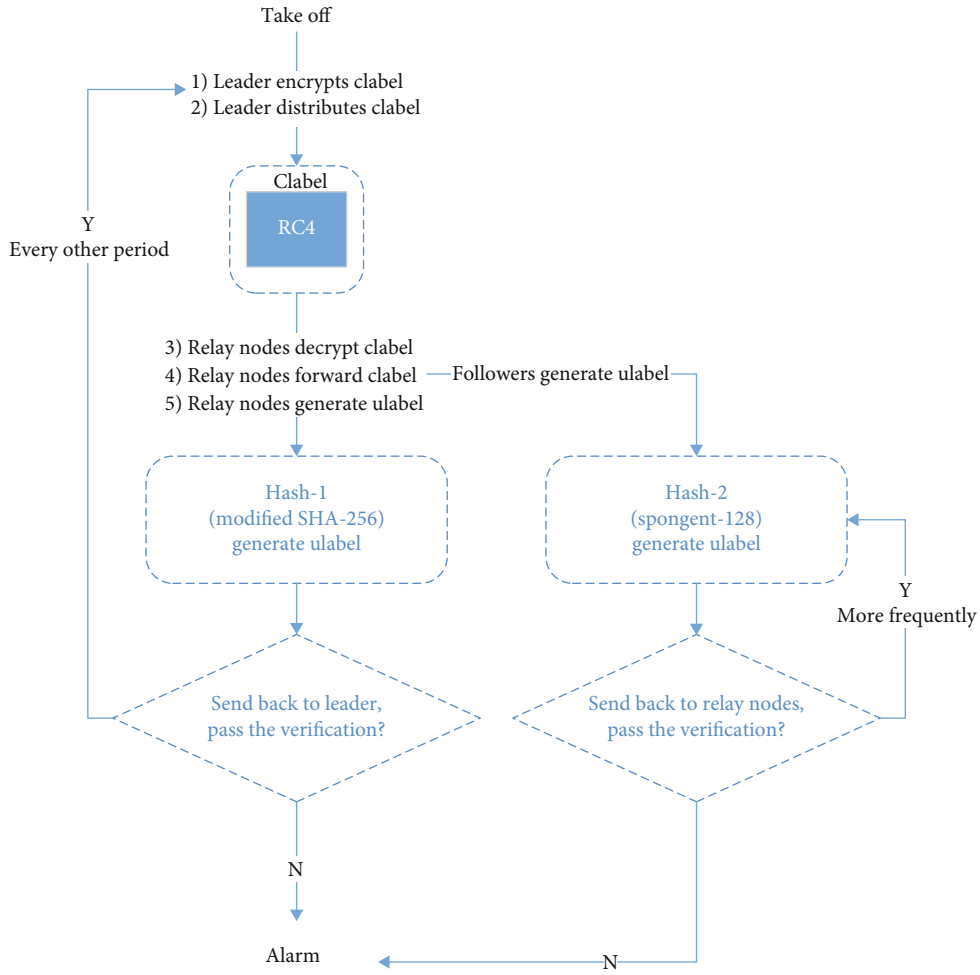


Figure 2: Mechanism workflow.

of the hash value and the number of calculation cycles on energy consumption and UAV, we use the SHA-256 hash algorithm for processing.

3.4. SPONGENT-128. In the existing Internet of Things (IOT) applications, a hash function with a collision resistance of 280 is acceptable to reduce overhead to the greatest extent in a safe environment. According to the different construction methods used in the function design, the design principles of the lightweight hash functions that have been publicly published can be divided into three categories: (1) based on permutation functions, (2) based on block ciphers, and (3) based on mathematical difficulties. We use a lightweight PRESENT-alike block cipher permutation function SPONGENT-128 [30] to solve the ulabel generation by high-frequency verification between the followers.

The domain extension structure used by SPONGENT-128 is the sponge structure, which can compromise encryption speed, memory requirements, and security by adjusting parameters [31]; the internal transformation uses an 8-round block cipher transformation. Through the three stages of initialization, absorption, and squeezing, five different output length example applications can be realized through different combinations of rate and capacity. The sponge structure is mainly composed of three components: (1) filling function P, (2) memory state S, and (3) transfer function F. Among them, the memory state S has a total of b bits, including two parts, the data rate r and the capacity c, and has an output of n bits ($n = c$). We set the parameters $r = 8$, $c = 128$, $n = 128$, and $b = 128$.

4. Random Label and Lightweight Hash-Based Security Authentication Mechanism Design

The previous section proposes several important models and definitions in the security encryption model of UAV swarms based on random labels. This section discusses their specific design schemes and implementations based on these definitions.

Considering that each UAV is an agent with computing and decision-making capabilities, the UAV can be considered the result of multiagent calculation and collaboration. After takeoff, the leading UAV acts as a mobile ground station and needs to perform tasks such as information collection and command, and safety is particularly important. In the entire process of performing the task, the task of safety certification mainly includes two aspects: one is the label generation and verification between the relay node and the peripheral member UAV and the other is the member UAV and the leader UAV label generation and verification. In order to ensure

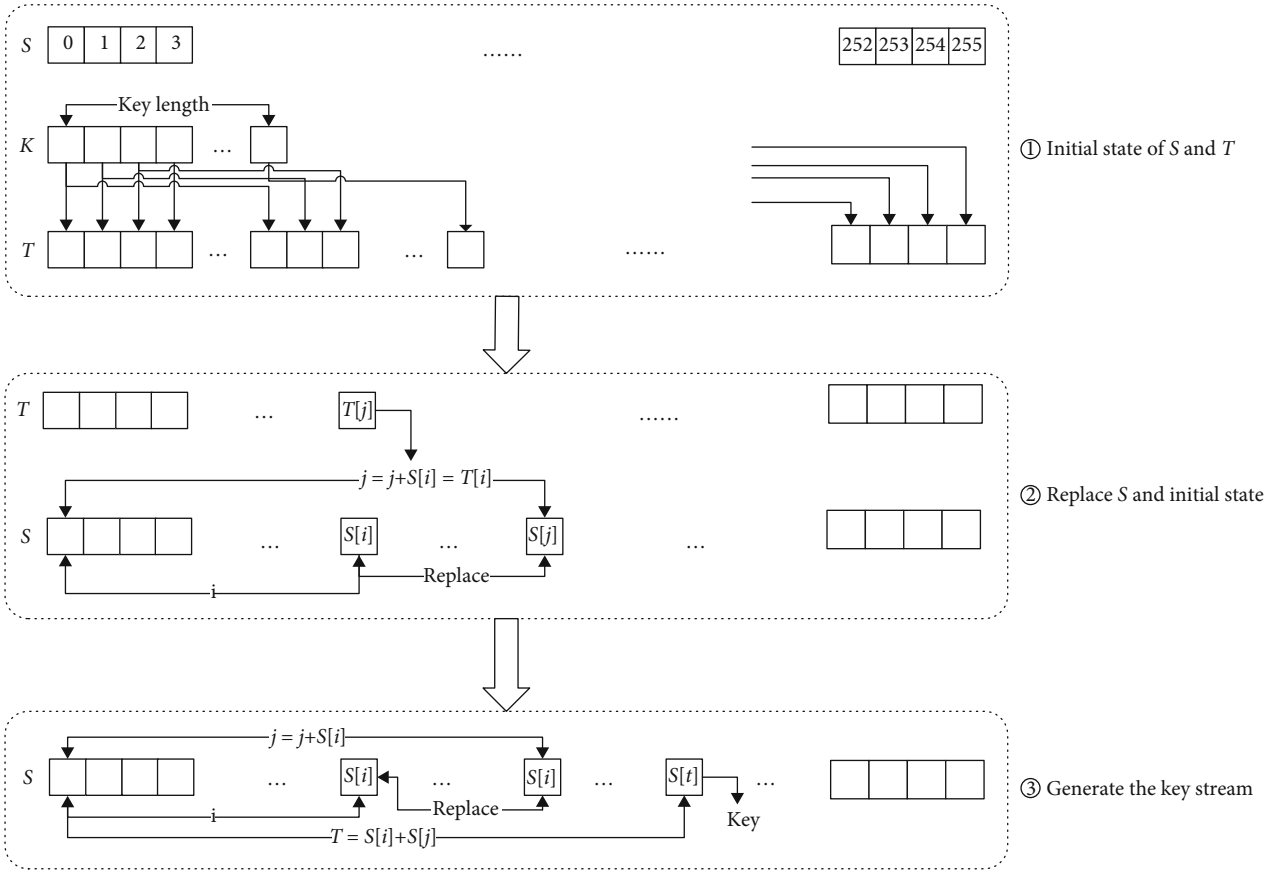


Figure 3: The process of RC4 encryption.

the safety of the leading UAV and to comply with the UAV's energy consumption limits and computing capabilities, two solutions have been specially introduced to ensure the feasibility of this model. The flow of this mechanism is discussed in Figure 2.

4.1. Encrypt the Certain Labels. In this part, we elaborate on the label processing logic of the UAV swarm system under the security encryption model based on random labels. After taking off, the leading UAV will distribute a clabel to followers in order to ensure the integrity of the surrounding UAVs and track their location. However, once the clabel is intercepted by the enemy or the leading UAV is once attacked, the entire system falls short. Besides, the label runs through the whole process of completing the task of the entire system. Therefore, in order to reduce the probability of the enemy attacking the leading UAV and also reduce the probability of the enemy intercepting the certain label, we firstly perform a lightweight encrypt algorithm to enhance the security of UAV data packets before issuing. Figure 3 is the detailed process of generating an encrypted label.

4.1.1. Generate Key Stream Seed 1. Given a state vector S, initialize it and assign $0, 1, 2, \cdots, 254,255$ to each byte in ascending order.

4.1.2. Enter the Initial Key. The initial key is defined by ourselves as any combination of less than 256 bytes and then filled in circularly until 256 bytes are filled, and then, the final result obtained is defined as a vector T to generate seed 2.

4.1.3. Disrupt Initial Seed 1. Perform a replacement operation on the state vector S, starting from the 0th byte and executing 256 times to ensure that each byte is processed.

4.1.4. Generate Key Stream and Encrypt. After the leader encrypts the label, it starts distributing. After each member UAV receives it, they start decryption, because the lightweight encryption scheme we use is symmetric encryption, and the original text is the same after being XORed twice with the same key. Thus, knowing the key means knowing the way to the original text. Of course, the key will be changed regularly so that malicious ones will not be able to detect the key. Finally, the original text is obtained, so the encrypted clabel UAV only needs to use the key stream to XOR one more time to obtain the original text and then perform subsequent operations. The relay nodes forward the identification label to the follower UAVs because they are relatively trustworthy. The complete process of generating LDPC-based multiuser superimposed information ciphertext is shown in Algorithm 1.

4.2. Between Leader and Leader Surrounding Relay Nodes. We focus on the modified SHA-256-based algorithm adapted to label generation and verification between the leader and relay nodes in this part.

1: **Generate key stream seed 1:**
2: Given a state vector S, initialize it, and assign 0, 1, 2, ..., 254, 255 to each byte in ascending order;
3: **for**int $i = 0 ; i < 256 ; i + +$**do**
4: $S[i] = i$;
5: **end for**
6: **Enter the initial key:**
7: The initial key is defined as any combination of less than 256 bytes;
8: Fill in circularly until 256 bytes are filled;
9: The final result is defined as a vector T, which is used to generate seed 2;
10: **Disrupt the initial seed 1:**
11: Perform a replacement operation on the state vector S, starting from the 0th byte, and executing 256 times to ensure that each byte is processed;
12: $j = 0$;
13: **for**int $i = 0 ; i < 256 ; i + +$**do**
14: $j = (j + S[i] + T[i]) \bmod 256$;
15: swap$(S[i], S[j])$;
16: **end for**
17: **Generate key stream and encrypt:**
18: $i = 0, j = 0$;
19: **while** clabel.length− **do**
20: $i = (i + 1) \bmod 256$;
21: $j = (j + S[i]) \bmod 256$;
22: swap$(S[i], S[j])$;
23: $t = (S[i], S[j]) \bmod 256$;
24: $k = S[t]$;
25: clabel[] = clabel[]^k;
26: **end while**

Algorithm 1: The encryption algorithm of RC4.

In order to prevent the enemy from using the exhaustive method to crack our encryption method, we made a certain degree of improvement to SHA-256—intercept a piece of data in the final message digest Hn and discard it, and then, fill the discarded data with a random function, and the whole process would not change the number of bits after SHA-256 encryption. The difficulty of cracking continues to increase without causing excessive energy consumption.

Under this model, the UAV data packet is enhanced through two main processes of label generation and label verification. The generation and verification of the label are highly coalesced, and only the labeled data packets need to be subjected to label verification. By setting the unused bit area to zero by default, we can distinguish marked data packets from unmarked data packets, so as to determine which data packets need to be verified. Data packets that do not require verification indicate that the UAV does not carry clabel when it departs from the ground station, and the system can directly trigger an alarm. Below is the detailed process of label generation and verification.

4.2.1. Label Generation. To generate a ulabel for the swarm boils down to the hashing process. In order to simplify the configuration, we can use the same hash parameter on the UAV. We let Pkt^{clabel} represent the data packet header with clabel and Sample(Pkt^{clabel}) the bit marked in it, and the length of which is represented by a 64-bit value. The ulabel to be generated by the UAV is formulated as follows:

$$\text{ulabel} = \text{Hash}\left(\text{Sample}\left(Pkt^{\text{clabel}}\right)\right), \tag{1}$$

where Hash$(\cdot)$ is the adopted SHA-256 algorithm. We adapt the SHA-256 algorithm to generate the ulabel for the packet header of UAV. First, we preprocess the Sample(Pkt^{clabel}), which is to add the needed information after the message to be hashed; second, we generate the massage list and use logical functions to calculate the message digest. It takes four steps to completely generate a ulabel: (a) initiating constants, (b) adding filling bit and length value, (c) calculating the message digest (the ulabel), and (d) modification.

4.2.2. Initiating Constants. In this algorithm, we use 8 initial hash values and 64 hash constants. The 8 initial hash values forming the initial mapping value H_0 are shown in Figure 4.

These initial values are the first 32 bits of the decimal part of the square root of the first 8 prime numbers (2, 3, 5, 7, 11,13,17,19) in natural numbers. For example, the decimal part of $\sqrt{2}$ is approximately 0.414213562373095048, and

$$0.414213562373095048 \approx 6 \times 16^{-1} + a \times 16^{-2} + 0 \times 16^{-3} + \cdots. \tag{2}$$

So, the first 32 bits of the decimal of $\sqrt{2}$ is $0x66a09e667$, that is, the $h0$. The 64 hash constant values are shown in Figure 5.

1	h1	:=	0×6a09e667
2	h2	:=	0×bb67ae85
3	h3	:=	0×3c6ef372
4	h4	:=	0×a54ff53a
5	h5	:=	0×510e527f
6	h6	:=	0×9b05688c
7	h7	:=	0×1f83d9ab
8	h8	:=	0×5be0cd19

Figure 4: The hash initial value of SHA-256.

Similar to the 8 initial hash values, these constants are derived from the first 32 bits of the decimal part of the cube root of the first 64 prime numbers (2, 3, 5, 7, 11,13,17,19,23,29 ⋯) in natural numbers.

4.2.3. Add Filling Bit and Length Value. Add filling bits to the end of the Sample(Pkt^{clabel}) so that the remainder of the Sample(Pkt^{clabel}) length is 448 after modulating 512. The first additional bit is 1 and then 0, until the length of 512 is sufficient and the remainder is 448. Then, we get Sample $(Pkt^{\text{clabel}})_1$. After that, we append Sample(Pkt^{clabel}) with Sample$(Pkt^{\text{clabel}})_1$, which gives us the message.

4.2.4. Calculating the Message Digest (ulabel). Since the minimum calculation unit of the algorithm is a 32-bit word, we break the message into sixteen 32-bit big-endian words $W0$, $W1, \cdots, W15$ and other 48 words $W16, W17, \cdots, W63$, which are derived from the following iterative formula, respectively:

$$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-2}) + W_{t-16}. \quad (3)$$

For each iteration, as shown in Figure 6, the 8 letters ABCDEFGH are updated according to certain rules, and the dark blue squares are 6 nonlinear logic functions. For the convenience of expression and calculation, we will express these words as x, y, z, and the functions are shown below.

$$\begin{aligned} \mathrm{Ch}(x,y,z) &= (x \wedge y) \oplus (x \wedge z), \\ \mathrm{Ma}(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), \\ \sum_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x), \\ \sum_1(x) &= S^6(x) \oplus S^{11}(x) \oplus S^{25}(x), \\ \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus S^3(x), \\ \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus S^{10}(x). \end{aligned} \quad (4)$$

The red blocks stand for the operation of mod 2^{32} addition, which means when the adding result is greater than 2^{32}, you must divide it by 2^{32} and find the remainder. The initial values of ABCDEFGH are $H_{i-1}(0), H_{i-1}(1), H_{i-1}(2), \cdots, H_{i-1}(7)$. For each iteration, let Kt denote the key over the t-th iteration, corresponding to the 64 constants we mentioned before, and Wt is the t-th word this block generates. The original message is cut into fixed-length 512-bit blocks; for each block, 64 words are generated, and the eight letters ABCDEFGH are cyclically encrypted by repeatedly running the cycle n times. The eight characters generated in the last loop are the hash string Hn corresponding to the nth block, which is the last 256-bit message digest. The above algorithm of the iteration process is presented in Algorithm 2.

1	428a2f98	71374491	b5c0fbcf	e9b5dba5
2	3956c25b	59f111f1	923f82a4	ablc5ed5
3	d807aa98	12835b01	243185be	550c7dc3
4	72be5d74	80deb1fe	9bdc06a7	c19bf174
5	e49b69c1	efbe4786	0fc19dc6	240ca1cc
6	2de92c6f	4a7484aa	5cb0a9dc	76f988da
7	983e5152	a831c66d	b00327c8	bf597fc7
8	c6e00bf3	d5a79147	06ca6351	14292967
9	27b70a85	2e1b2138	4d2c6dfc	53380d13
10	650a7354	766a0abb	81c2c92e	92722c85
11	a2bfe8a1	a81a664b	c24b8b70	c76c51a3
12	d192e819	d6990624	f40e3585	106aa070
13	19a4c116	1e376c08	2748774c	34b0bcb5
14	391c0cb3	4ed8aa4a	5b9cca4f	682e6ff3
15	748f82ee	78a5636f	84c87814	8cc70208
16	90befffa	a4506ceb	bef9a3f7	c67178f2

Figure 5: The hash constant value of SHA-256.

4.2.5. Modification

(1) Use the interception function to intercept the encrypted ciphertext (Hn), and intercept the number ($0 <$ number < 32) digits from the beginning number ($0 <$ benninnumber < 32) position to obtain the password A, where $A = \text{left}(\text{SHA} - 256(\text{password}), \text{beginnumber} - 1)$

(2) Use the intercept function to intercept the value B of the number digits of the encrypted plaintext, where $B = \text{right}(\text{SHA} - 256(\text{clabel}), \text{SHA} - 256 - \text{digit} - (\text{beginnumber} + \text{number} - 1))$

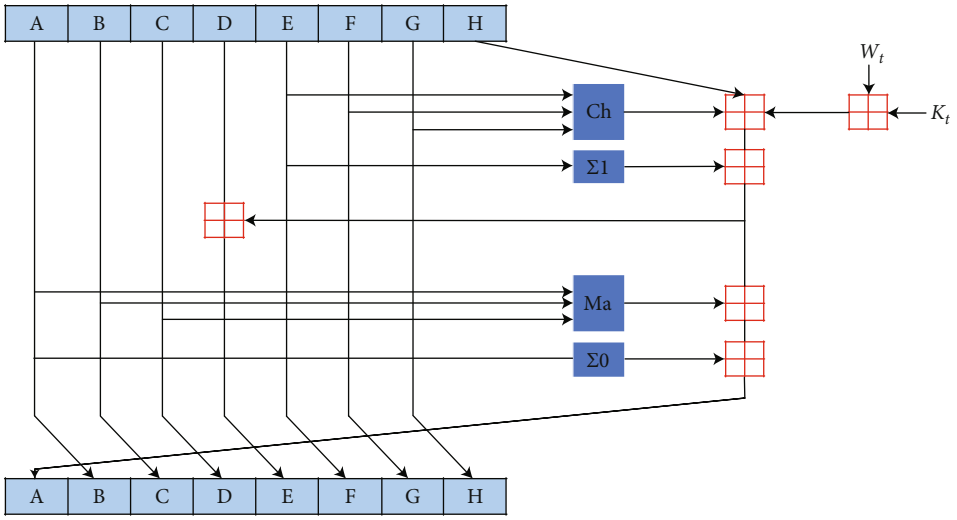


Figure 6: The iteration process of SHA-256.

1: **for***i* from 0 to 63**do**
2: **Logical function operation:**
3: S0 = (A rightrotate 2) ^(A rightrotate 13) ^(a rightrotate 22);
4: Ma = (A and B) ^(A and C) ^(B and C);
5: t2 = S0 + Ma;
6: S1 = (E rightrotate 6) ^(E rightrotate 11) ^(E rightrotate 25);
7: Ch = (E and F) ^((not E) and G);
8: $t1 = h + S1 + Ch + K[i] + W[i]$;
9: **Update each word:**
10: H=G;
11: G=F;
12: F=E;
13: E = D + t1;
14: D = C;
15: C = B;
16: B = A;
17: A = t1 + t2;
18: **Add the hash output of the message block to the existing hash output:**
19: h0 = h0 + A;
20: h1 = h1 + B;
21: h2 = h2 + C;
22: h3 = h3 + D;
23: h4 = h4 + E;
24: h5 = h5 + F;
25: h6 = h6 + G;
26: h7 = h7 + H;
27: **Output the final hash value (big-endian):**
28: digest = hash = h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7;
29:**end for**
30: **return** digest;

Algorithm 2: The iterative algorithm of SHA-256.

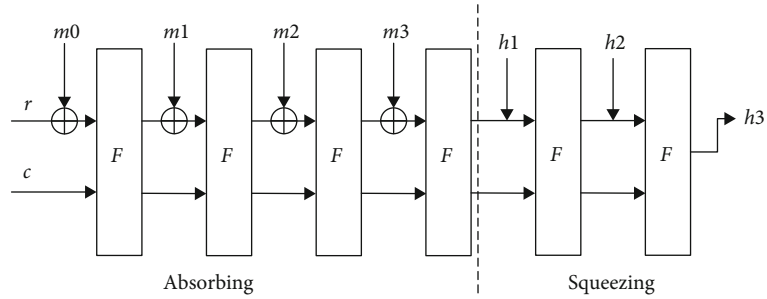


Figure 7: Absorbing and squeezing phases of spongy structure.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

Figure 8: The state of S-box.

(3) Use the random function $\text{gen}_\text{k}\text{ey}(\text{number})$ to fill in the value of the intercepted number; the converted password value is $\text{encrypt}_\text{p}\text{assword} = A\&\text{get}_\text{k}\text{ey}(\text{number})\&B$

From the above, we get the final message digest (ulabel). This process randomly hashes the input to the output, which makes reverse engineering using input-output pairs difficult.

4.3. Label Verification. The authentication between the leader and relay nodes will seem successful as long as the following two conditions are met: first, the data packet comes from the correct front-hop node (UAV) and second, the data packet should carry the correct ulabel generated by the UAV.

(1) The leader UAV verifies whether the clabel of the member UAV is correct, and if correct, it indicates that the data packet comes from the correct previous hop node

(2) The leader UAV also generates an Hn according to the received clabel and then intercepts the first half from the beginning number to get A', and the second half gets B' and then reads the A and B parts of the password from the database, and finally if $A = A'$ and $B = B'$, it is considered that the password entered by the user matches the password in the database, indicating that $\text{ulabel}' = \text{ulabel}$, and the verification is passed. Otherwise, the label verification fails and the UAV triggers an alarm

4.4. Among Followers. We focus on the lightweight SPONGENT-128-based algorithm adapted to label generation and verification among the followers in this part.

4.4.1. Label Generation. After the follower UAV receives the clabel encrypted by the leader, it first decrypts the RC4 to obtain the original clabel. Because the lightweight encryption scheme we use is symmetric encryption, the original text is obtained by XORing the plaintext with the same key twice, so the encrypted clabel UAV only needs to use the keystream to XOR once again to get the original text.

The encryption process of the hash function using the sponge structure can be finished through three stages [30, 32]:

Initialization Phase. The b-bit memory state S is initialized to all 0 s, and the plaintext message clabel is filled with the filling function P. As shown in Figure 7, the specific method is to first add 1 after the message, and then, add a sufficient number of 0 s, so that the number of bits of the message after filling is exactly r an integer multiple (for example, when the message has 57 bits and $r = 8$, the filled content should be "1000000").

Absorbing Phase. The filled message will be divided into r message blocks (mi). The message block is XORed to the rightmost r-bit position of state S, which forms part of the hash output. Each time the r-bit message is XORed with the first r-bit of S, then the entire S is converted by F to obtain the new state and then repeat the above steps (absorption), until all the filled messages are absorbed.

F conversion: $F_2^b \longrightarrow F_2^b$. It is an R-round block cipher conversion of the input b-bit state. $s\text{BoxLayer}_b$ and $p\text{Layer}_b$ describe how the state evolves. $l\text{Counter}_b(i)$ is the state of the LFSR at time i that depends on b. It generates an integer constant for round i and adds it to the rightmost bit of the state. $l\text{Counter}_b(i)'$ is the value of $l\text{Counter}_b(i)$, whose bit order is reversed and added to the leftmost bit of the state. Generally speaking, $s\text{BoxLayer}_b$ means to use a 4-bit to 4-bit S-box: $F_2^4 \longrightarrow F_2^4$, distributed and operated $b/4$ times. S-box (substitution-box) is the basic structure of the symmetric key algorithm [33] to perform substitution calculation. The state of S-box is shown in Figure 8.

Besides, $p\text{Layer}_b$ means to move j in S state to position $Pb(j)$:

$$P_b(j) = \begin{cases} \dfrac{j \cdot b}{4 \bmod b - 1}, & \text{if } j \in \{0, \cdots, b-2\}, \\ b - 1, & \text{if } j = b - 1. \end{cases} \tag{5}$$

Table 1: The ability of SPONGENT-128 against differential attack.

Rounds	ASN	Maximum difference probability
5	10	2^{-22}
10	29	2^{-68}

*l*Counter*b* is one of the three $\log_2 R$ bit linear feedback shift registers (LFSR). The initial value assigned to the register is called the "seed", because the operation of LFSR is deterministic, so the data flow generated by the register is completely determined by the current or previous state of the register. Moreover, since the state of the register is limited, it will definitely end up in a repeated loop. However, through primitive polynomials, LFSR can generate seemingly random sequences with very long cycles. Shift registers are simple in structure and fast in operation. Most practical key stream generators are based on shift registers, and shift register theory has become the basis of modern stream cryptosystems. In this scheme, the LFSR is recorded every time the *S* state is used, and its final value is 1. Let ζ denote the unit root in the corresponding binary finite field. The SPONGENT-128 we use has the original trinomial $\zeta^7 + \zeta^6 + 1$ 7-bit LFSR, which is initialized to "1111010."

Squeezing Phase. After all the messages are absorbed, the first *r* bits of *S* are output. If the number of digits to be output is $n > r$, then continue to perform *F* transformation on *S* and then output the first *r* bits of *S* and so on until the total number of digit output reaches *n* bit.

4.4.2. Label Verification. Because the followers are semi-trusted to us, we need to authenticate the follower at a higher frequency. This authentication is verified by the relay node according to SPONGENT-128 to verify the ulabel returned by the follower. If the result is the same as the returned one and if they are consistent, the follower is credible. If they are inconsistent, the relay node will trigger an alarm.

In summary, after taking off, the leader first performs RC4 encryption to the clabel and then distributed them to the relay nodes, and the relay nodes forward to the peripheral member UAV followers to ensure the security of the database; after the relay node decrypts the clabel, they adapt SHA-256 to generate ulabels and are verified by the leader every certain period to ensure the safety of relay nodes during the flight; after receiving the forwarded clabels, followers adapt SPONGENT-128 to generate ulabels to finish the verification cycle faster, and the ulabel sent back to the relay node for verification indicates that its identity guarantees the safety of the peripheral member UAVs; the security of the information in the wireless transmission process is protected through two hash schemes, and there is no too much additional energy consumption.

Therefore, our proposed scheme enables the inside of the UAV swarm to use labels that may be unknown to the attacker to mark the data packet header. In addition, the labeling strategy is synchronized between the leading UAV and the following UAV for data packet verification. If a malicious UAV attacker manipulates a follower UAV, it must forge the correct label to avoid verification.

4.5. Safety Analysis. This part analyzes the security model proposed in this paper from mainstream attack methods such as traditional attacks and differential attacks.

4.5.1. Security Assumptions. Cryptography has defined three security assumptions about hash functions. If a hash function satisfies the following three assumptions, such a hash function can be considered safe [34]:

(i) The original image having stable assumption: choose output value h arbitrarily, and make $H(M) = h$ by finding a message string M which is infeasible

(ii) The second preimage firm assumption: given a message string M, find another message string Z to make $H(M) = H(Z)$ which is infeasible

(iii) Assumption of collision stability: it is infeasible to find any two message strings M and Z such that $H(M) = H(Z)$.

4.5.2. Traditional Attack. From the perspective of attack principles, traditional attacks do not use the structure of the hash function and any weak algebraic properties and are only affected by the length of the hash value. Therefore, the most effective way to resist traditional attacks is that the hash value must have sufficient length. The SHA-256 used in this paper uses a 256-bit hash value, and its magnitude is $2128 \approx 3.4 * 1038$, and it takes several million years to find a collision [7].

According to the literature [30], for the verification between the relay node and the UAV, it can be seen that the domain extension structure of the sponge structure has good safety performance in a large-scale and lightweight environment. Therefore, this paper adopts the domain extension structure of SPONGENT-128 verifying the integrity between UAVs and analyzing the cost of mainstream attack types to break this scheme:

(1) The range of costs to be paid for a collision attack is

$$\min\left\{2^{n/2}, 2^{c/2}\right\}. \tag{6}$$

(2) The range of costs to be paid for the preimage attack is

$$\min\left\{2^{n}, 2^{c}, \max\left\{2^{(n-R)/2}, 2^{c/2}\right\}\right\}. \tag{7}$$

(3) The range of costs to be paid for the second preimage attack is

$$\min\left\{2^{n}, 2^{c/2}\right\}. \tag{8}$$

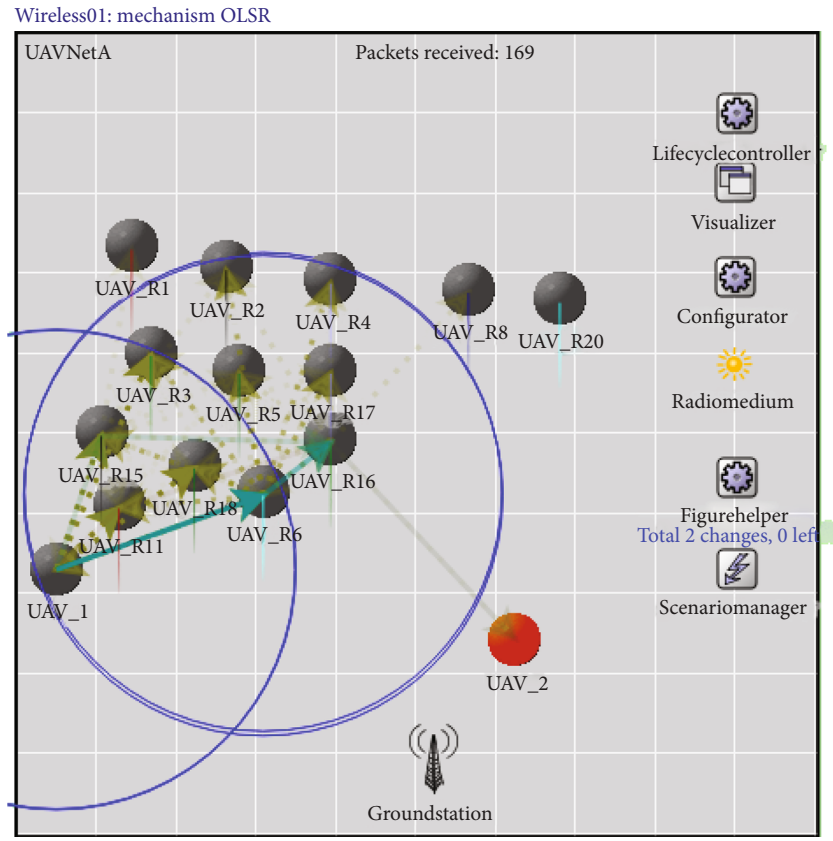


Figure 9: OMNeT++ simulation interface.

Set the main parameters in the sponge structure: output digits $n = 128$, memory state digits $b = 136$, capacity $c = 128$, data rate $r = 8$, and output digits in the extrusion stage $R = 8$, so the solution in this paper resists traditional attacks. The capabilities are as follows:

(i) Collision attack resistance: 264

(ii) Preimage attack resistance: 2120

(iii) The second preimage attack resistance: 264

4.5.3. Differential Attack. Differential attack is currently one of the most effective methods to decipher the iterative hash function. The basic method is to use the impact of the input difference of the plaintext on the output difference and use the high probability of inheritance or elimination of the difference to produce the final same output [35]. The safety of a hash function ultimately depends on whether the overall collision of the function can be found. Because the SHA-256 algorithm has an iterative structure, according to the avalanche effect of the iterative algorithm, as the number of rounds increases, the corresponding overall collision complexity will be sharply ascending; this makes it very difficult to find the overall collision. Until now, the existing attacks have not been able to find a SHA-256 overall collision. Through the analysis of the Chabaud-Joux attack on SHA-256, a partial collision of SHA-256 was found, with a complexity of 266 [7], but an overall collision of SHA-256 could not be found, so the SHA-256 algorithm can also resist existing differential attacks.

For a hash function with a sponge structure, its ability to resist differential attacks is directly determined by its internal transformation. In order to find the upper bound of the internal transformation's ability to resist differential attacks, the general method used is to find the number of minimum differential active S-boxes (ASN) in the encryption process. For the SPONGENT-128 used in this paper, the experimental results of the proponent of the scheme cited in this paper [30] are shown in Table 1.

5. Simulation Experiment and Result Analysis

In view of the fact that there is no platform that supports UAV swarm simulation, we tested and analyzed a variety of popular simulation platforms and chose OMNeT++ as the UAV flight simulation test environment. OMNeT++ is a modular, component-based C++ simulation library and framework, mainly used to build network simulators. OMNeT++ can be used for free in noncommercial simulations such as academic institutions and teaching. OMNeT++ itself is a simulation framework, and there is no model for network protocols such as IP or HTTP. There are several external frameworks for the main computer network simulation models. The most commonly used one is INET, which provides various network protocols and technologies of various models, such as IPv6 and BGP.

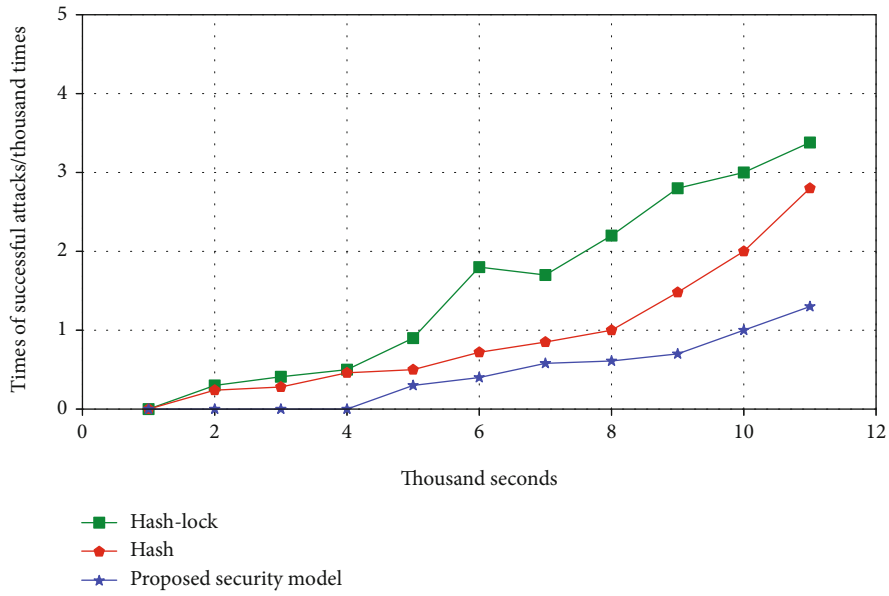


Figure 10: The times of successful attacks.

The UAV simulation test platform is composed of three main modules: communication module, computing module, and mobile module. The communication module is based on the INET framework, in which the physical layer uses the radio model and the medium model, the data link layer uses the ad hoc-based 802.11 protocol, the routing protocol uses the OLSR (Optimized Link State Routing) protocol, and the transmission protocol uses the UDP protocol. As shown in Figure 9, the calculation module performs related calculations and the operation of the two hash algorithms based on the information obtained by various sensors and provides different functions for the cluster nodes. For example, each node simulates the equipped geographic information sensor by reading its coordinate location information. The mobile module is based on the mobile model of INET and has been redesigned and implemented according to the UAV formation requirements. This module can simulate the waypoint flight mode of the UAV.

We compare the security model based on the hash chain [36] and the security model based on the hash-lock protocol [37] with the scheme proposed in this paper in terms of throughput, delay, and security to verify the scheme proposed in this paper on security and performance. In order to better evaluate and compare the security of these protocols, an attack program is added here to simulate an attacker's attack on the protocol. This paper simulates several common network attacks. We call the generation and authentication functions in the file/hash.cc. OMNeT++ processes the arriving data packets by calling the generation function in ProcessPacket(·) and forwards the processed data packets by calling the verification function in PacketCallback(·). The following is an analysis of the simulation results.

5.1. Statistics of the Number of Successful Attacks. With 1000 seconds as a time node, the number of attacks of the three models at each time node is counted, as shown in Figure 10. It can be seen that the security model based on the hash-lock and hash chain is almost linearly distributed, basically every unit of time can be successfully attacked, and the security scheme based on the hash-lock protocol is based on the security of the hash chain after 4 s. After 8 s, the number of successful attacks on the scheme has increased at a faster rate, indicating that as long as the attack continues for a long time, these two schemes have great security risks.

The encryption scheme proposed in this paper, with the increase in time, has basically remained stable in the probability of successfully resisting common attacks. Compared with the other two schemes, the curve trend is relatively flat and there is no obvious growth trend. Therefore, the ability to resist common attacks always maintains advantages and has good security. Moreover, in the first four seconds of the simulation, the scheme proposed in this paper can completely resist the attack behavior, and it is quite effective for the UAV cluster with high real-time and mobility to resist the invasion of attackers.

5.2. Throughput and Delay. We compare the delay and throughput of the solution that does not use any security solution, the security model based on the hash chain, the security model based on hash-lock, and the solution that uses the security model of this paper.

As shown in Figure 11, for the throughput, it can be seen that when the number of packet lengths is small (64, 128 bytes), the security scheme has a greater impact on throughput, because the label length accounts for a larger proportion of the packet length with a security label. So it will reduce the throughput of the system, but when the packet length is longer (256, 512, and 1024), the throughput of the four schemes is not much different, so the security scheme is more suitable for data packets containing more data. The application scenario is the scenario with a large packet length, and the UAV will carry a large amount of data. Therefore, the security scheme proposed in this paper does not have much impact on the throughput while ensuring the security of the

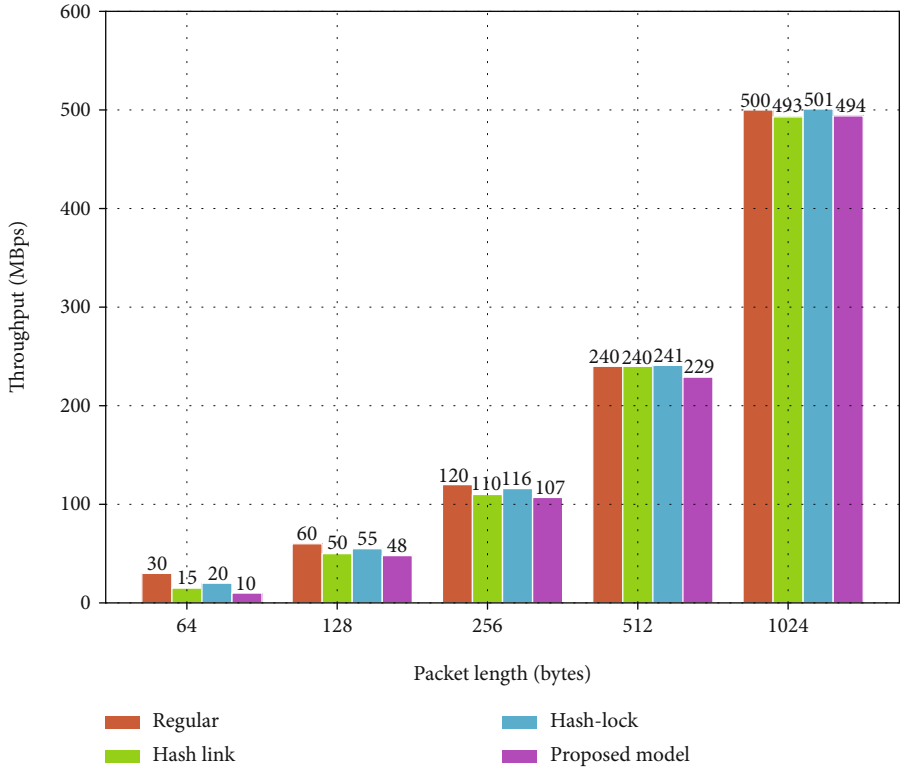


Figure 11: Throughput.

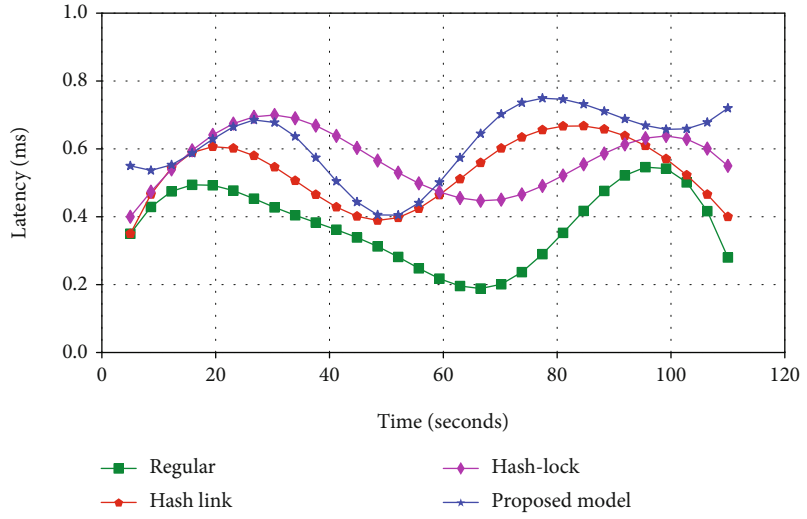
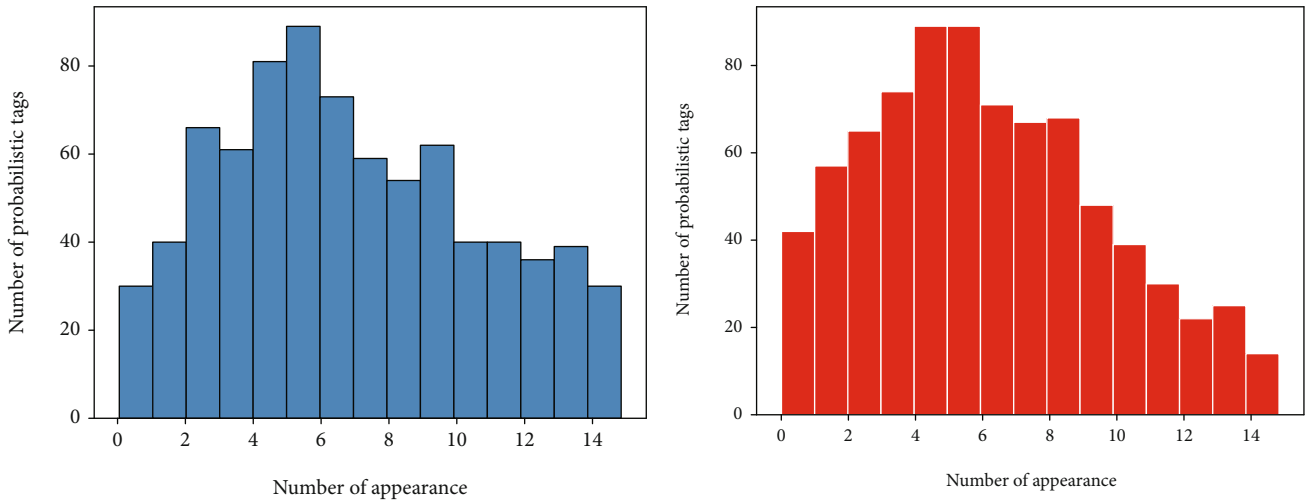


Figure 12: Delay.

system. As shown in Figure 12, for the delay, in order to facilitate observation and analysis, we ignored some inevitable jitter and focused on mainstream delay. As shown in the figure, the average delay of the security model based on the hash chain and hash-lock is higher than that of the solution proposed in this paper, and the delay of the security model proposed in this paper is not much different from that of the normal mode, only an increase of about 0.15 ms. From this aspect, it also illustrates the feasibility and low latency of the program.

5.3. Feasibility. While improving the security and robustness of the system, we also examine the distribution of uncertain labels, which significantly reduces the possibility of an attacker accurately predicting uncertain labels through random guessing. Because if the distribution is biased towards certain labels, the attacker is likely to use these labels to improve the prediction success rate. From Figure 13, we can see that the uncertain labels generated by the UAV have approximately normal distribution, which well limits the range of the attacker's random inference.



(a) Hash-1 (b) Hash-2

Figure 13: The distribution of ulabel generated by hash-1 and hash-2.

We used a 10-digit uncertain label and considered 5000 other packets in the same stream. We expect the average number of labels to appear as 5000/210 = 4.9. When this expected value is reached and the number of occurrences is 4, the above two distributions reach their peaks. The above results show that the model proposed in this paper will not cause the attacker to continue to speculate on our label, nor will it issue wrong labels. After all, whenever it detects a packet with an incorrect label, it will alert the system. In addition, constantly refreshing the custom key also helps invalidate the attacker's inference.

6. Conclusions

Based on the previous work, our mechanism greatly reduces statistical inference. The SHA-256 algorithm and SPONGENT-128 are used to generate the ulabel of the UAV message header, which ensures that the system does not increase a large amount of energy consumption, while the safety is greatly improved. Since the current UAV supports thousands of [38] rule updates per second, we regularly update the UAV's custom key and the label authentication rules with the leader.

Finally, we proposed the situation where an attacker invaded the UAV swarm performing military missions and proposed a random labeling mechanism solution. This mechanism enables the UAV swarm system to mark packet headers that are difficult for attackers to infer and guess. The labeling strategy is to conduct swarm authentication through collaboration across the leader UAV, relay nodes, and member UAVs. If an attacker manipulates a damaged UAV to try to enter the system, it must forge the correct label to avoid verification, and our mechanism restricts the attacker to random guesses. In addition, this paper uses OMNeT++ to implement a security authentication mechanism based on random labels for large-scale UAV swarms. The experimental results show that, in terms of throughput, delay, feasibility, etc., the scheme proposed in this paper can achieve accurate inspection with reasonable cost and ensure the safety of the system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This manuscript is an extension of [8], published in the 17th IEEE International Symposium on Parallel and Distributed Processing with Applications (IEEE ISPA 2019).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China, under Grant 2019YFB2102002;in part by the Key Research and Development Program of Jiangsu Province, under Grant BE2019012; and in part by the National Natural Science Foundation of China, under Grant 62001217.

References

[1] C. Zhong, J. Yao, and J. Xu, "Secure UAV communication with cooperative jamming and trajectory control," *IEEE Communications Letters*, vol. 23, no. 2, pp. 286–289, 2019.

[2] P. Perazzo, F. B. Sorbelli, M. Conti, G. Dini, and C. M. Pinotti, "Drone path planning for secure positioning and secure position verification," *IEEE Transactions on Mobile Computing*, vol. 16, no. 9, pp. 2478–2493, 2017.

[3] J. Zhang, T. Chen, S. Zhong et al., "Aeronautical ad~hoc networking for the Internet-above-the-clouds," *Proceedings of the IEEE*, vol. 107, no. 5, pp. 868–911, 2019.

[4] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1123–1152, 2016.

[5] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: a communications viewpoint," *IEEE Communications Surveys Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[6] B. Song, J. Y. Hwang, and K. Shim, "Security improvement of an RFID security protocol of ISO/IEC WD 29167-6," *IEEE Communications Letters*, vol. 15, no. 12, pp. 1375–1377, 2011.

[7] D. Sun and Y. Mu, "Security of grouping-proof authentication protocol for distributed RFID systems," *IEEE Wireless Communications Letters*, vol. 7, no. 2, pp. 254–257, 2018.

[8] L. Liu, H. Qian, and F. Hu, "Random label based security authentication mechanism for large-scale UAV swarm," in *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 229–235, 2019.

[9] H. Sedjelmaci, S. M. Senouci, and N. Ansari, "A hierarchical detection and response system to enhance security against lethal cyber-attacks in UAV networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 9, pp. 1594–1606, 2018.

[10] Z. Birnbaum, A. Dolgikh, V. Skormin, E. O'Brien, and D. Muller, "Unmanned aerial vehicle security using recursive parameter estimation," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 692–702, 2014.

[11] R. Cabaniss, V. Kumar, and S. Madria, "Multi-party encryption (MPE): secure communications in delay tolerant networks," *Wireless Networks*, vol. 21, no. 4, pp. 1243–1258, 2015.

[12] T. O. Olwal, K. Djouani, and A. M. Kurien, "A survey of resource management toward 5G radio access networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1656–1686, 2016.

[13] B. Luo, X. Li, J. Weng, J. Guo, and J. Ma, "Blockchain enabled trust-based location privacy protection scheme in VANET," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2034–2048, 2020.

[14] Y. Yapici, İ. Güvenç, H. Dai, and A. Bhuyan, "Physical layer security for UAV swarm communications via protected zone," in *Resilience Week (RWS)*, vol. 1, pp. 174–177, IEEE, 2019.

[15] Y. Zhou and L. Li, "A trust-aware and location-based secure routing protocol for WSN," *Applied Mechanics and Materials*, vol. 373-375, pp. 1931–1934, 2013.

[16] E. Kline, A. Afanasyev, and P. Reiher, "Shield: DoS filtering using traffic deflecting," in *19th IEEE International Conference on Network Protocols*, pp. 37–42, 2011.

[17] T. Hayajneh, R. Doomun, G. Al-Mashaqbeh, and B. J. Mohd, "An energy-efficient and security aware route selection protocol for wireless sensor networks," *Security and Communication Networks*, vol. 7, no. 11, p. 2038, 2020.

[18] R. Sangeetha and M. Yuvaraju, "Secure energy-aware multipath routing protocol with transmission range adjustment for wireless sensor networks," in *IEEE International Conference on Computational Intelligence and Computing Research*, pp. 1–4, 2012.

[19] J. J. Lotf, M. Hosseinzadeh, and R. M. Alguliev, "Hierarchical routing in wireless sensor networks: a survey," in *2010 2nd international conference on computer engineering and technology*, vol. 3, pp. V3650–V3654, IEEE, 2010.

[20] X. Chen, K. Makki, K. Yen, and N. Pissinou, "Sensor network security: a survey," *IEEE Communications Surveys Tutorials*, vol. 11, no. 2, pp. 52–73, 2009.

[21] T. Hu and Y. Fei, "Qelar: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 6, pp. 796–809, 2010.

[22] N. Kabir and S. Kamal, "Secure mobile sensor data transfer using asymmetric cryptography algorithms," in *International Conference on Cyber Warfare and Security (ICCWS)*, pp. 1–6, 2020.

[23] A. Broumandan, A. Jafarnia-Jahromi, V. Dehghanian, J. Nielsen, and G. Lachapelle, "GNSS spoofing detection in handheld receivers based on signal spatial correlation," in *IEEE/ION Position, Location and Navigation Symposium*, pp. 479–487, 2012.

[24] P. Y. Montgomery, T. E. Humphreys, and B. M. Ledvina, "Receiver autonomous spoofing detection: experimental results of a multi-antenna receiver defense against a portable civil GPS spoofer," in *Proceedings of the 2009 International Technical Meeting of The Institute of Navigation*, vol. 1, pp. 124–130, 2009.

[25] P. Gope and T. Hwang, "A realistic lightweight authentication protocol preserving strong anonymity for securing RFID system," *Computers & Security*, vol. 55, pp. 271–280, 2015.

[26] K. Bu, Y. Yang, Z. Guo, Y. Yang, X. Li, and S. Zhang, "FlowCloak: defeating middlebox-bypass attacks in software-defined networking," in *IEEE INFOCOM 2018- IEEE Conference on Computer Communications*, pp. 396–404, 2018.

[27] Z. A. Qazi, R. Miao, C. C. Tu, V. Sekar, L. Chiang, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Acm Sigcomm Conference on Sigcomm*, pp. 27–38, 2013.

[28] E. Khan, M. W. El-Kharashi, F. Gebali, and M. Abd-El-Barr, "Design and performance analysis of a unified, reconfigurable HMAC-Hash unit," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 12, pp. 2683–2695, 2007.

[29] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen message attack ∗," *Discrete Algorithms and Complexity*, vol. 17, no. 2, pp. 287–310, 1987.

[30] A. Bogdanov, M. Kneˇzevi'c, G. Leander, D. Toz, K. Varıcı, and I. Verbauwhede, "SPONGENT: A lightweight hash function," *Lecture Notes inComputer Science*, vol. 6917, pp. 312–325, 2011.

[31] P. Megha Mukundan, S. Manayankath, C. Srinivasan, and M. Sethumadhavan, "Hash-One: a lightweight cryptographic hash function," *IET Information Security*, vol. 10, no. 5, pp. 225–231, 2016.

[32] P. Li, L. Han, X. Tao et al., "Hashing nets for hashing: a quantized deep learning to hash framework for remote sensing image retrieval," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 10, pp. 7331–7345, 2020.

[33] N. Meghanathan, B. K. Kaushik, and D. Nagamalai, "Advances in networks and communications," *Communications in Computer & Information science*, vol. 132, 2012.

[34] M. Wang and Y. Li, "Hash function with variable output length," in *2015 International Conference on Network and Information Systems for Computers*, pp. 190–193, 2015.

[35] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, 1991.

[36] S. Chang, Y. Park, and B. B. Ashok Babu, "Fast IP hopping randomization to secure hop-by-hop access in SDN," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 308–320, 2019.

[37] O. Changqing, J. Wu, L. Zhengyan, and H. Shengye, "An enhanced security authentication protocol based on hash-lock for low-cost RFID," in *2008 2nd International Conference on Anti-counterfeiting, Security and Identification*, pp. 416–419, 2008.

[38] G. Lombardi, E. Medvet, and A. Bartoli, "A language for UAV traffic rules in an urban environment and decentralized scenario," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 1139–1143, 2017.