Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 8913910, 11 pages
https://doi.org/10.1155/2019/8913910

Research Article

UAV-Undertaker: Securely Verifiable Remote Erasure Scheme with a Countdown-Concept for UAV via Randomized Data Synchronization

Sieun Kim,[1] **Taek-Young Youn**,[2] **Daeseon Choi**,[3] **and Ki-Woong Park**[4]

[1]*SysCore Lab., Sejong University, Seoul, Republic of Korea*
[2]*Electronics and Telecommunications Research Institute, Deajeon, Republic of Korea*
[3]*Dept. of Medical Information, Kongju National University, Kongju, Republic of Korea*
[4]*Dept. of Computer and Information Security, Sejong University, Seoul, Republic of Korea*

Correspondence should be addressed to Ki-Woong Park; woongbak@sejong.ac.kr

Received 31 January 2019; Revised 28 March 2019; Accepted 22 April 2019; Published 16 May 2019

Academic Editor: Ilsun You

Unmanned aerial vehicles (UAVs) play an increasingly core role in modern warfare, with powerful but tiny embedded computing systems actively applied in the military field. Confidential data, such as military secrets, may be stored inside military devices such as UAVs, and the capture or loss of such data could cause significant damage to national security. Therefore, the development of securely verifiable remote erasure techniques for military devices is considered a core technology. In this study, we devised a verifiable remote erasure scheme with a countdown-concept using randomized data synchronization to satisfy securely verifiable remote erasure technology. The scheme allows the GCS (Ground Control Station) to remotely erase data stored in the UAV, even on loss of communication, and returns proof of erasure to GCS after erasure. Our approach classifies the accumulated data stored in the UAV as a new data type and applies the characteristics of that data type to generate the proof of erasure. We select a small-volume data sample (rather than all of the data) and perform prior learning only on that sample; in this way, we can obtain the probative power of the evidence of erasure with a relatively small amount of traffic. When we want to erase data of 100 Mbytes of remote device, 100 Mbytes of data transfer is required for related work, whereas our system has data transfer according to the ratio of amount of randomly selected data. By doing this, communication stability can be acquired even in unstable communication situations where the maximum traffic can change or not be predicted. Furthermore, when the UAV sends the proof of erasure to the GCS, the UAV does its best to perform the erasure operation given its situation.

1. Introduction

In 2011, the US military lost control of an American Lockheed Martin RQ-170 Sentinel unmanned aerial vehicle (UAV) during a mission over Afghanistan. The UAV was captured by Iranian forces [1], which successfully extracted information, released a video obtained from the captured UAV, and subsequently constructed a copy of the RQ-170, the Saegheh [2]. As demonstrated by this episode, the theft of military UAVs causes significant damage, including the loss of secret data, such as collected information [3]. Consequently, it is indispensable to assure the security of the UAV [4–6].

Erasing data as needed can safeguard confidential information stored on remote devices. Generally, if a user sends an erasure request, the remote device receives it and performs the erasure. However, UAVs remotely performing a mission cannot always receive the signal [7–9]. As such, techniques are needed to completely erase the data stored in remote UAVs following a loss.

Recognition that sensitive information has been leaked (i.e., the erasure process has not been performed) is critical, and so confirmation of erasure must be performed in a verifiable manner (verifiable erasure). Verifying the result of the erasure operation as 1-bit response has low reliability [10].

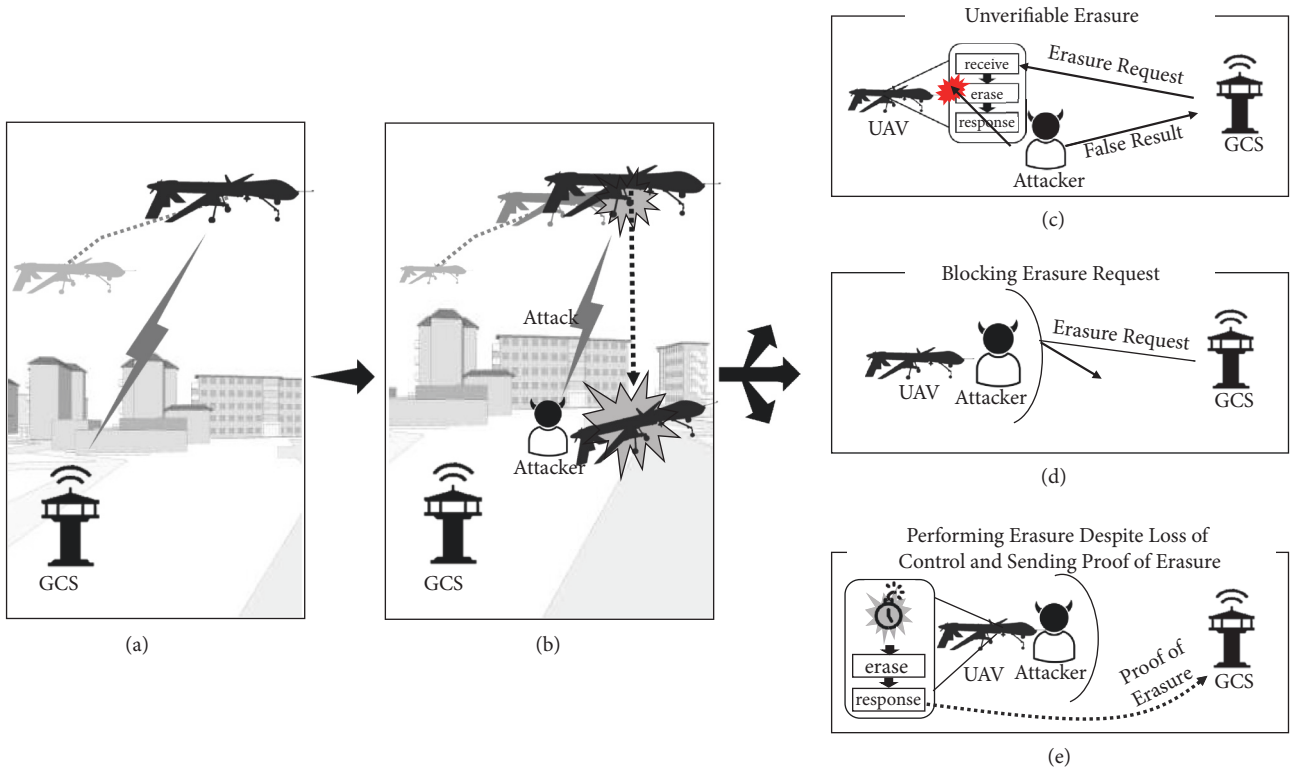

Figure 1: Three situations faced by unmanned aerial vehicles (UAVs) after attack or capture. (a) UAV on a flight mission; (b) UAVs performing a flight mission is captured by an attacker; (c) receiving unverifiable result of erasure; (d) blocked erasure requests by attackers; and (e) performing erasure operation and transmitting proof of erasure following capture.

Obvious proof of erasure must be presented, and there must be no mechanism by which malicious users can generate false evidence.

Figure 1 indicates three situations that UAVs may be faced with after attack or capture. In conventional remote erasure systems [11, 12], a Ground Control Station (GCS; that is, the control center that provides human control of UAVs) can erase data by sending an erasure signal to a lost UAV. However, when communication is cut, conventional remote erasure is not available. We devised counter-based erasure, which can erase data in an uncontrollable environment. This study is an extension of our previous work [13, 14], in which we focused on the system design of trustworthy remote erasure for UAVs. However, our objective here was to devise a mechanism to select random seed data using accumulated data and develop a method for trustworthy remote erasure. The devised scheme has a count value, which decrements value one by one and erases data when the counter value reaches zero. The GCS can retain stored data only through periodically sending the specific value to the UAV. If the GCS does not send the specific value to the remote UAV and the counter reaches zero, the UAV erases the data. In terms of verification, most existing mechanisms provide nothing or only a simple response (erase or not erase). Our remote erasure system provides relatively complex but verifiable proof of erasure through randomized data synchronization. Upon a process of erasure of data, the remote UAV continuously generates a proof of erasure until the UAV is completely stopped; this proof is repeatedly sent to the GCS. The GCS checks that the proof has been received and verifies that remote data are erased.

The remainder of this paper is organized as follows: Section 2 reviews the related works of verifiable remote data erasure. Section 3 describes the materials and gives details of our mechanism UAV-Undertaker. Section 4 describes two experiment results about overhead in terms of communication and computation. This paper ends in Section 5 with conclusion on our work.

2. Related Works

To resolve problem of erasure of remotely stored data with verifiability, the researchers have devised approaches which generate a provable value of erasure operation. We analyzed several of them and identified the limiting factors in applying a verifiable erasure for UAVs.

Perito and Tsudik [15] use a PoSE (Proof of Secure Erasure) for secure code updates on embedded devices by verifying the erasure of memory. A verifier sends verifier-selected randomness of the size of memory to a prover, and the prover's memory is filled with the received value. The prover returns the very same randomness written on the memory to the verifier. As both the verifier and the prover send a random value of memory size to each other, this

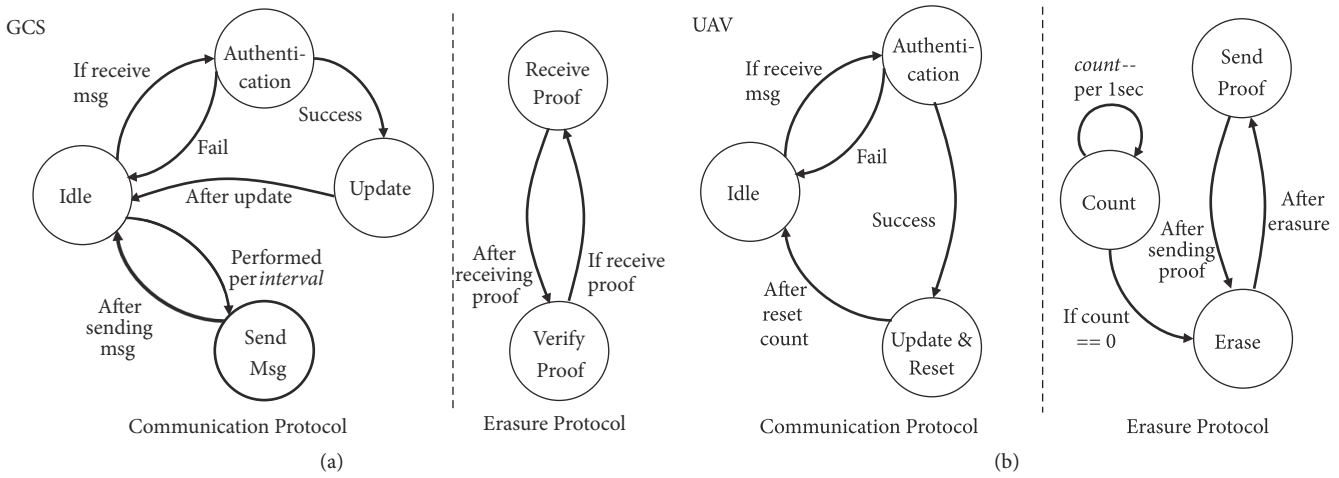

Figure 2: Overall process of the proposed remote erasure system. (a) Unmanned aerial vehicle (UAV)-Undertaker from the Ground Control Station (GCS) viewpoint; (b) UAV-Undertaker from the UAV viewpoint. If the count value becomes zero, the UAV erases data and repeatedly sends the proof of erasure to the user (i.e., the GCS).

protocol has a huge overhead concerning communication. In our system, as shown in Figure 9 of Section 4, when a small amount of data (i.e., α =2) is selected as a seed and transmitted, the instantaneous amount of transferred data is significantly lower than when live tracking all data (i.e., α =100). In other words, our system shares the burden of the required data transfer amount to verify the erasure operation. Therefore, our system can significantly reduce instantaneous amount of transferred data which is a limitation of the above study. Dziembowski et al. [16] proposed a scheme that reduces the communication complexity of PoSE. A verifier sends a seed to a prover. The prover performs a hash function using the seed and retains the sizable result value using all of the prover's memory. The calculated hash value can only be generated once because it uses a secret key stored in the memory, and so it can be proof of erasure. However, while this scheme reduces the communication overhead of PoSE, a huge calculative overhead arises. Ammar et al. [17] introduced SPEED, which guarantees erasure on embedded system. The protocol is implemented in isolated memory using the Trusted Software Module. A verifier's distance is measured by the Distance Bounding (DB) protocol, which establishes an upper bound on the physical distance between a verifier and a prover. The prover authenticates the verifier through the DB protocol and erases data if verification is passed successfully. To verify erasure, the prover sends the proof as the message authentication code value of the entire memory. However, the scheme is limited in terms of distance and so is not applicable for UAVs.

Our approach solves verifiable erasure through proof of erasure [15–18] that only the user can identify, as in the above studies. However, our approach creates a proof of erasure in such a way that the memory blocks each have a dependency on the block, which guarantees more robust evidence. Furthermore, in contrast to past studies, which did not consider losses in the communication environment, we aimed to achieve remote erasure of UAVs after hijack, loss, and/or disconnection. Also, the sensor system in cloud is similar to our system in that it collects data from remote devices and processes them to achieve specific goals. However, in the case of the cloud sensor system, the collected information is limited to the sensor data and is merely used for providing the monitoring to the user. In the case of our system, the collected information is the memory information of the remote device. It can be seen that there is a difference from the cloud sensor system in that it verifies whether or not the erasure operation is performed by utilizing this.

3. Proposed System Architecture

Our UAV-Undertaker consists of two parts: the communication protocol and verifiable erasure protocol. We confirm whether or not control of the UAV is lost by communicating with the UAV. Therefore, it is important to verify that a message originates from the stated sender (i.e., authentic communication) and has not been changed. This authentication role is implemented as the communication protocol using a hash chain mechanism [19]. We also implement the verifiable erasure protocol, which generates proof of erasure to verify that the erasure operation was really performed. We classify the data region according to the number of times the data will be written in order to increase the probative power of proof of erasure with just a small amount of data (thus, improving efficiency). The operation result value is created by accurately performing the implemented erasure operation. Therefore, a cryptographic accelerator, such as the TPM (Trusted Platform Module), must be used in order to make the results of the computation trustworthy.

Figure 2 shows the overall process of our approach from both the GCS and UAV viewpoints. From the GCS viewpoint, the GCS sends an authentication request message to the UAV at *interval*s and then waits for a message from the UAV. If the GCS receives a message from the UAV, it verifies that the message is authentic; if so, it updates the value to be

Table 1: Notations of the entity and messages.

Definition of the Entity Symbols	
GCS	Ground Control Station
UAV	Unmanned Aerial Vehicle
Denotation	
msg = $\alpha \parallel \beta$	Message contains two contexts (α and β)
msg = $\alpha \parallel (\beta)$	Message always contains α context, but β context is optionally contained.
Definition of the Message Symbols	
$K_{x,y}$	A symmetric key shared between x and y
$E\{K, B\}$	Encrypt B with key K
x	Value to compute with hash function
n	Number of hash function calculations
r	Retransmission bit
i	Authentication count
$h(x)$	Result of x in the hash function
Nonce	Generated random data against replay attacks
hl	Address of hot data modified after previous authentication
al	Address of accumulated data modified after previous authentication
hv	Set containing the addresses of hot data randomly selected and the values of those data
av	Set containing the addresses of accumulated data randomly selected and the values of those data
PoE	Proof of Erasure
Definition of the Greek Symbols	
α	Ratio of amount of randomly selected data in accumulated data region
β	Ratio of amount of randomly selected data in hot data region
γ	Number of re-authentication permits
δ	Initial value of the counter

sent next. If communication with the UAV is cut off and the UAV performs the erasure operation and the GCS repeatedly receives the proof of erasure from the UAV and verifies it. From the UAV viewpoint, after decrementing the counter by one, the UAV waits for a message from the GCS (idle state). A UAV that has successfully authenticated a GCS update of the hash value for the next authentication, initializes the decreasing count value to δ (i.e., the initial value of the counter), and waits for the message to be received again (idle state). If the UAV does not receive a message from the GCS before the count value reaches a certain value, the UAV repeatedly erases stored data and repetitively sends proof of erasure through a low frequency.

3.1. Communication Protocol. To allow the sender of each message to be authenticated, hash chains and message encryption are used in the communication protocol. To encrypt a message, the UAV must share the symmetric key securely offline with the GCS before departure. Our approach is designed to execute the erasure protocol after γ times of requests for reauthentication, because the UAV may be confronted with a mere transient communication failure, where γ can be set freely according to the circumstances of mission. Table 1 defines the entity symbols and the message symbols.

Figure 3 shows the message flow when communication between the UAV and the GCS is performed without problems. First, the GCS encrypts the number of hash function calculations (n) and the value to compute with hash function (x) using a symmetric key already shared between the GCS and the UAV; it then sends an initialization request message (*Message 1-1*) containing that encrypted value to the UAV. When the value is received from the GCS, the UAV computes $h^n(x)$, encrypts it, and sends an initialization response message (*Message 1-2*) to the GCS. The GCS confirms that the UAV received the correct x and n. The GCS and the UAV, who have verified each other, start exchanging authentication messages using a hash chain. The GCS sends messages to the UAV at regular intervals. When the maximum count value is δ (seconds) and the number of reauthentication times is γ, the *interval* is calculated using (1). The constant "1" in (1) is added because GCS waits for the *interval* and then sends the next message. The constant "2" in (1) is multiplied to prevent the timer from being initialized during the maximum authentication latency that can occur on our system.

$$interval = \frac{\delta}{2(1+\gamma)} \ (seconds) \tag{1}$$

After the UAV receives a message (*Message 1-3*) containing the hash value from the GCS, the UAV calculates the hash value of the received value and compares the value with the previously stored value. If two values are equal, the UAV initializes the count value to δ and updates the stored value with the hash value received from the GCS.

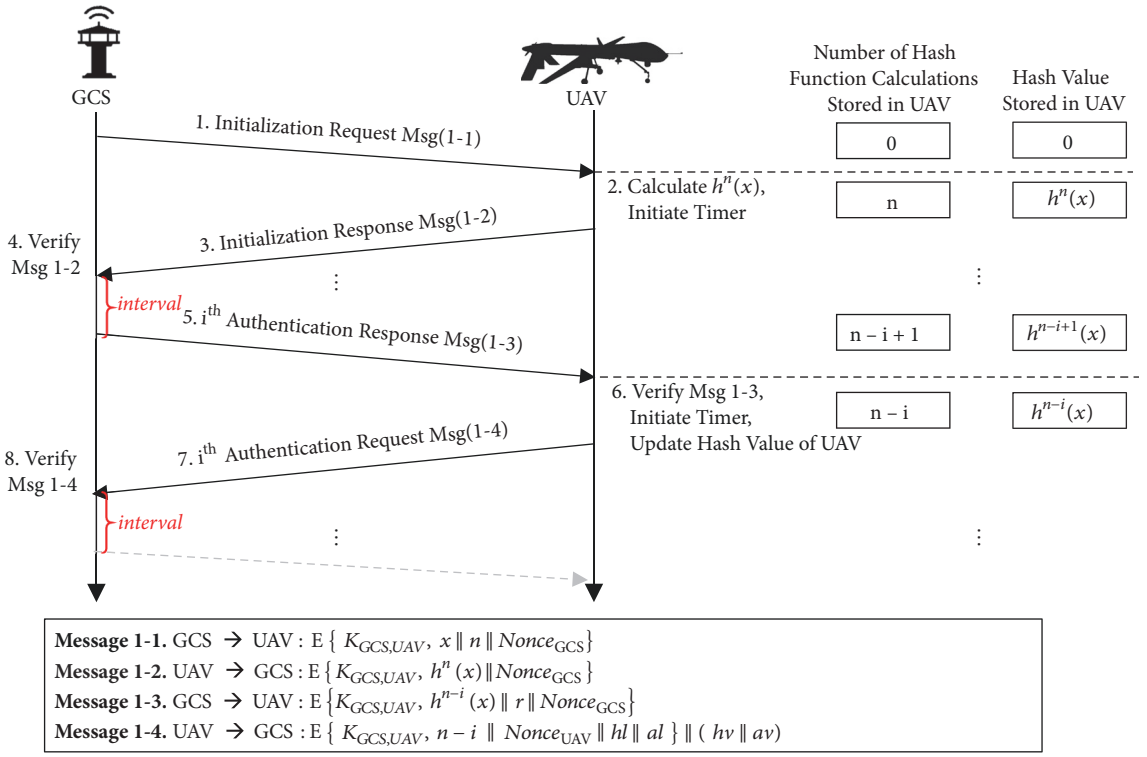

Figure 3: Message flow depending on the communication environment. Messages 1-1–1-4 are those sent when communication between the unmanned aerial vehicle (UAV) and the Ground Control Station (GCS) is performed without problems.

After confirming that the message is sent from the GCS, the UAV sends a message (*Message 1-4*) containing the memory information along with the n − i to the GCS. *hv* and *av* (changed memory information) contained in *Messages 1-4* are not encrypted and transmitted. The reason is that, in our system, *hv* and *av* can include very large amounts of data and encrypting these data is expected to have a very high computational overhead. Even if the malicious user successfully acquires the unencrypted *hv* and *av*, the user cannot generate the false proof because the last hash value used in the erasure operation cannot be determined. These transactions are repeated *n* times if communication is good. When the number of authentication (*i*) exceeds *n* (i.e., the maximum number of hash function calculations), the GCS and the UAV reexchange the new *n* and *x* through *Message 1-1* and *Message 1-2*.

Figure 4 shows the message flow from when UAV could not receive the hash value from the GCS until the counter value reaches 0, at which point it initiates the erasure protocol. If the GCS sends an authentication request message (*Message 2-1*) to the UAV but does not receive the authentication response message from the UAV, the GCS waits the *interval* time and then sends again the message containing the same hash value. At this time, the *r* bit (i.e., the retransmission) is set to 1 and included in the message (*Message 2-2*). If the UAV faced a temporary communication failure, it will subsequently receive a reauthentication request message (*Message 2-2*) from the GCS; the UAV checks the *r* bit, recognizes resending, and confirms that the hash value stored in the UAV and received value are equal. If the two values are equal, the UAV returns value (n − i), which means the next hash value request. At that time, the UAV does not update the stored counter value or hash value. If the two values are not equal, the UAV calculates the hash value of the received value and compares it with the stored value. If the two values are equal, the UAV resets the counter value and updates the stored hash value. After that, the user and the UAV resume normal communication. However, if the next hash value is not received from the GCS and the value of the counter inserted in the UAV reaches 0, the UAV performs an erasure protocol and sends a message (*Message 2-3*) including proof of erasure and the memory information changed since the most recent successful authentication to the GCS through a certain frequency. In order for the GCS to verify that the UAV performed an erase operation, the GCS must know the changed memory information in the UAV. However, after the communication between the GCS and the UAV is lost, the GCS do not know information of data changed in the UAV. Thus, *Message 2-3* contains the changed memory information since the communication was disconnected. As our approach takes the best effort approach, the UAV repeatedly performs the erasure operation after transmitting the proof of erasure and continues to send the proof of erasure (*Message 2-4*) to the GCS. The number of times an erase operation is performed depends entirely on a given amount of time to the UAV after the erase operation has begun.

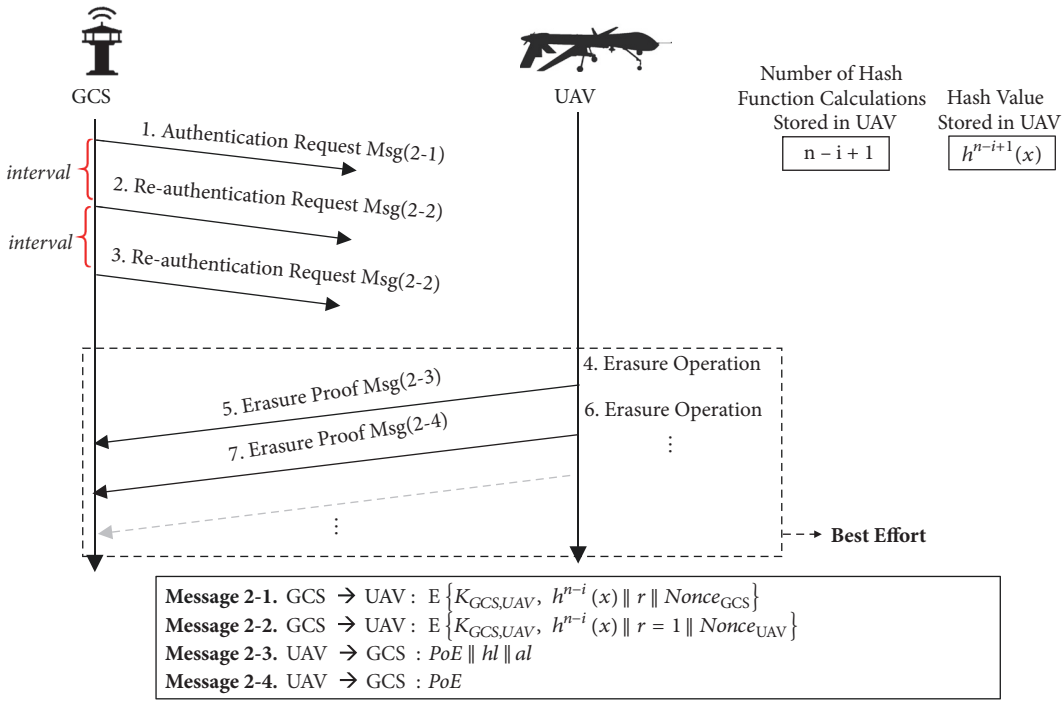

Figure 4: Message flow depending on the communication environment. If the unmanned aerial vehicle (UAV) does not receive the hash value from the Ground Control Station (GCS) before the counter value reaches 0, the UAV proceeds with the erasure process and sends proof of erasure to the GCS.

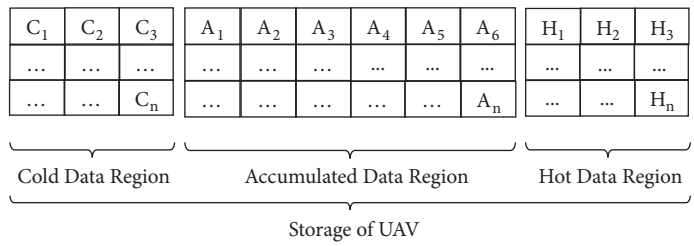

Figure 5: Unmanned aerial vehicle (UAV) storage divided into three regions: cold data region, accumulated data region, and hot data region.

3.2. Verifiable Erasure Protocol. For data erasure, our approach classifies the UAV storage into three data regions according to the expected number of writes of the data (Figure 5). The first region is a cold data region (i.e., data that will never be modified after the UAV departs), the second is the accumulated data region (i.e., data that will not be modified after it has been written; for example, video shot by the UAV), and the third is a hot data region (i.e., data that is actively modified). The mechanism assumes that the GCS knows the address values of the three data regions.

Our proposed proof of erasure generation method requires the live tracking of accumulated data and hot data (except for cold data that does not change because each data point has a dependency on each other in the proof of erasure). However, since tracking all the data causes considerable communication overhead, we randomly select data to act as seeds and transmit authentication messages containing just those data (Figures 6 and 7).

The selected data is managed by recording the address value and data type in the seed block table. In the accumulated data region, the number of seed data selects α% of the number of modified data blocks from time t_{n-1}, at which the previous seed block was selected to the current time t_n, at which the seed block selection occurs. Therefore, when selecting seed data, the seed is selected uniformly regardless of the time at which the data were generated. In the hot data region, the number of seed data selects *the hot data region size* $\times$ β % among modified data blocks from time t_{n-1}, at which the previous seed block was selected to the current time t_n, at which the seed block selection occurs. The value of α and β can be directly selected by the user. If the GCS and the UAV are in very good communication (high bandwidth) and there is no problem sending and receiving a lot of data; the GCS can increase the probative power of proof by setting the variable (α) high. Conversely, if the GCS and the UAV are in poor communication (low bandwidth) and cannot exchange

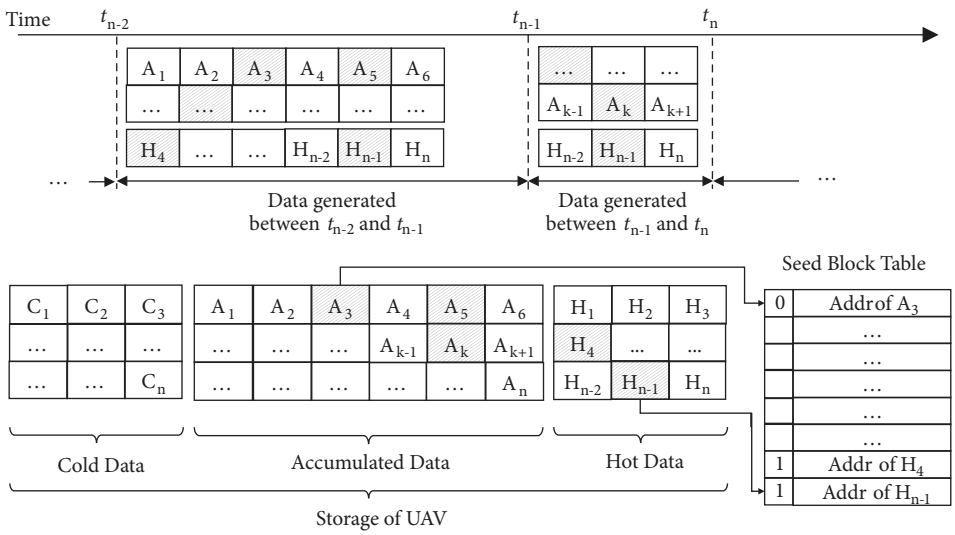

FIGURE 6: Random selection of data from unmanned aerial vehicle (UAV) storage and the seed block table. The gray block of memory indicates data selected as seed blocks. The seed blocks are selected from data generated between time t_{n-1}, at which the previous seed block was selected, and time t_n at which the seed block selection occurs.

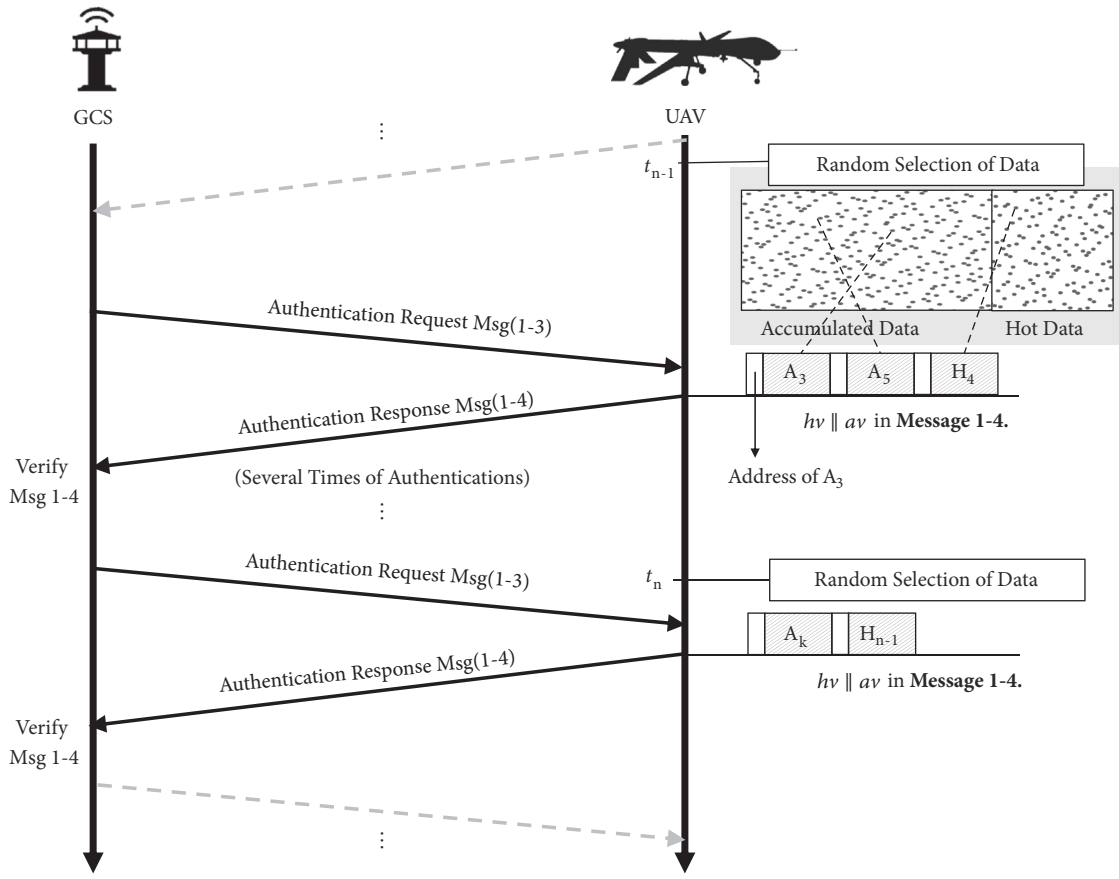

FIGURE 7: Process of sending randomly selected seed data to the Ground Control Station (GCS). The gray blocks represent randomly selected memory blocks, and t represents an arbitrary time.

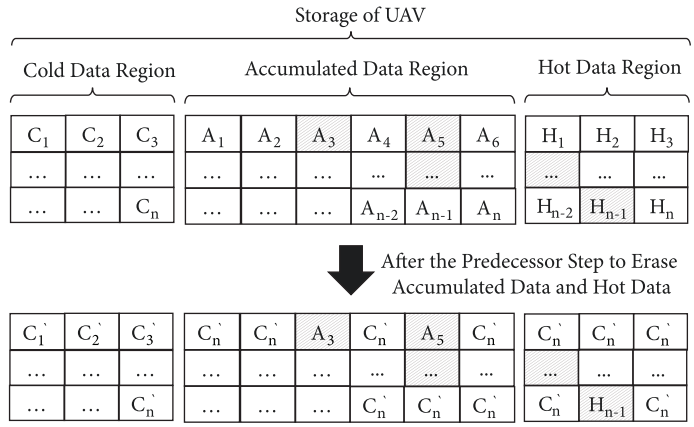

Figure 8: Memory state of the unmanned aerial vehicle (UAV) after the predecessor step to erase accumulated data and hot data. In accumulated data and hot data regions, the remaining blocks, excluding the seed data blocks, are overwritten with C_n', which is the result of the erasure operation of the cold data region.

large amounts of data, the GCS can lower the amount of communication data by setting the variable (α) very low. In other words, the GCS can set the variable according to the situation where the GCS use our system.

If the UAV counter reaches zero, the erasure protocol proceeds in the following order: erasure of cold data, erasure of accumulated data, and erasure of hot data. For erasure of cold data, the erasure operation is performed without any predecessor because the GCS already knows all of the values. The UAV overwrites the first block (C_1) of the cold data with the last hash value received from the GCS. The UAV performs an XOR operation of the C_2 block of the memory and C_1 block wiped with the last hash value received from the GCS and wipes the C_1 block with the calculated value. From now on, the wiped C_1 block is denoted by C_1'. Next, the C_2 block is wiped to the value obtained by the XOR operation of the C_1' block and the C_2 block. The repeated wiping of memory proceeds in the same way (see (2)) until wiping the C_n block (i.e., the last block of cold data).

$$C_n' = C_n \oplus C_{n-1} \tag{2}$$

For accumulated data region and hot data, the GCS cannot know all of the stored values and so predecessors are required. The address value of the changed data after the last successful authentication time should be recorded in the main memory. The remaining blocks, excluding the seed data block, are overwritten with C_n', which is the result of the erasure operation of the cold data. The bottom side of Figure 8 shows the memory state after this predecessor step. Then, the UAV performs the XOR operation with the A_2 block of accumulated data and the A_1 block of accumulated data and wipes the A_2 block with the result value. If confronted with the seed data (e.g., A_3) during the above operation, the UAV performs the XOR operation with the A_2' block of accumulated data and the hash value of the A_3 block of accumulated data and wipes the A_3 block with the result value (see (3)). In this way, all data stored in the accumulated data region and the hot data region are erased. In other words, the seed data of hot data are also calculated from (4).

$$A_n' = h(A_n) \oplus A_{n-1} \tag{3}$$

$$H_n' = h(H_n) \oplus H_{n-1} \tag{4}$$

The process above is defined as the first round. Starting from the second round, the erasure process proceeds without distinguishing the data region, unlike the first round. The UAV wipes C_1' with the result of the XOR operation of the C_1' block and H_n' block. After that, the UAV wipes C_2' with the result of the XOR operation of C_1'' block and C_2'. This process continues until the last block of hot data is overwritten, without any distinction of the data region, to complete the second round. To prevent forensic, we proceed through multiple rounds. We assume that the repeatedly wiped memory cannot be recovered, so it is impossible to extract information from the memory. After the second round process, as proof of erasure, the block H_n'' value is computed using a hash function, and the UAV sends that value (proof of erasure) to the GCS as Frequency Shift Keying (FSK) via the low frequency. The GCS who has received the proof of erasure from the UAV will be able to calculate the proof of erasure and confirm the validity of the message.

In the first part of the erasure process, the last hash value sent by the GCS is used as a seed in the erasure process to enhance security. The hash value used as the seed is an encrypted value, so an attacker cannot determine this value. Each value calculated by the above erasure process has a dependency on the previous memory data. Therefore, even if the attacker knows some blocks of memory, a false proof cannot be generated.

4. Experimental Results

To test performance of the devised protocol, we implemented an experimental environment using the *T2080* as the UAV flight computer [20]; the *T2080* has a 128 Mbytes NOR flash

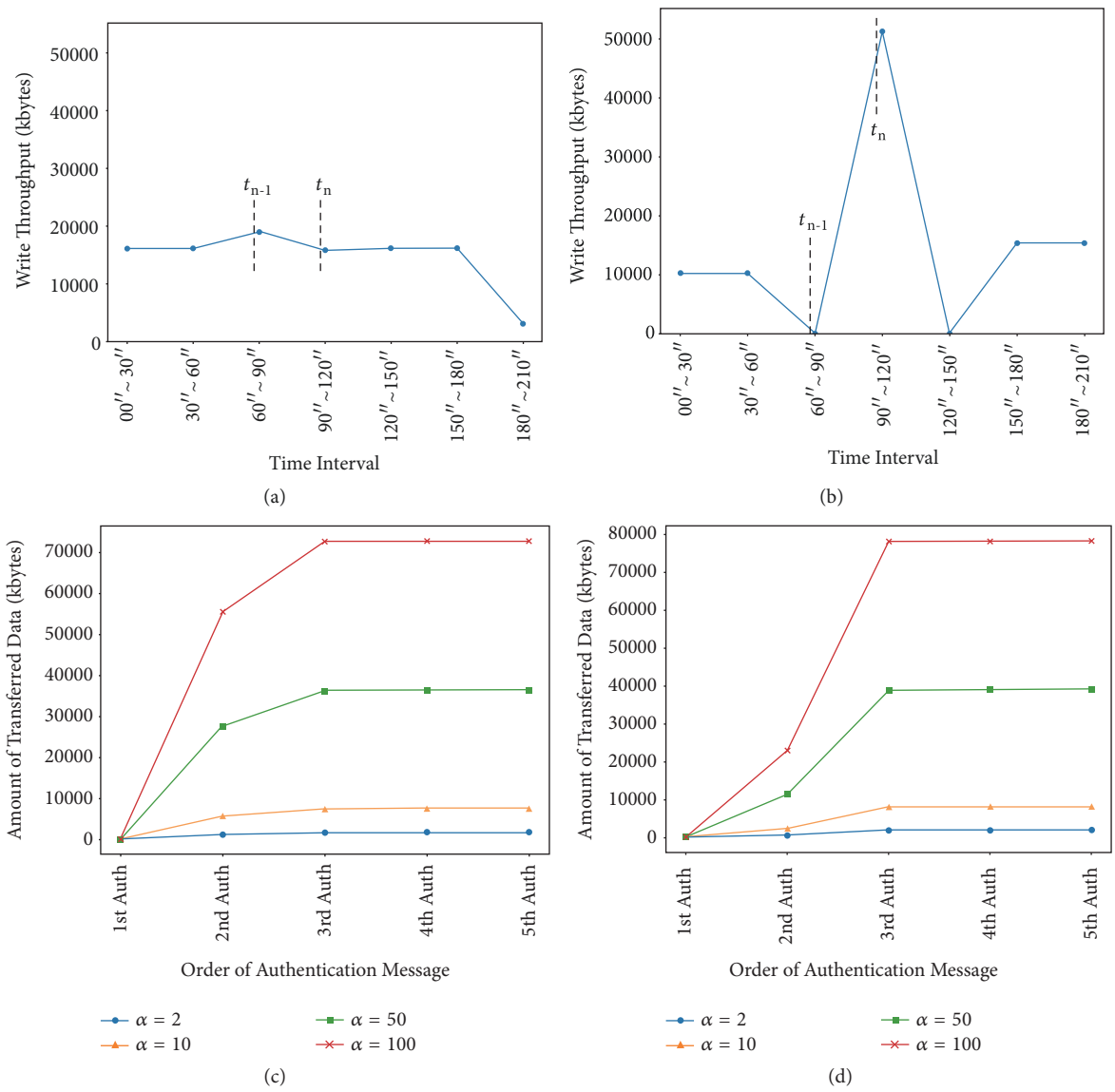

FIGURE 9: Amount of data transferred according to the rate at which accumulated data are generated and α. (a) Constant amount of data written to the SD card over time interval; (b) random amount of data written to the SD card over time interval; (c) amount of transferred data when the data is generated as shown in Figure 9(a) on the unmanned aerial vehicle (UAV); and (d) amount of transferred data when the data is generated as shown in Figure 9(b) on the UAV.

memory, SD connector to interface, and SATA interface. Through this implementation, we conduct two experiments to investigate how practical the proposed scheme is: the communication overhead according to α and the erasure latency of the erasure process according to memory size. We conducted experiments on SD cards inserted in the *T2080*. The block size used in the XOR operation was determined based on how many 512-byte blocks were sequentially read at a time and the size of one block used in our experiment was 1Kbytes, which was determined by reading 512-byte blocks twice in succession.

4.1. Communication Overhead Experiment. This experiment measured the amount of data transferred according to the rate at which accumulated data are generated and α. The amount of transferred data in the experiment is the amount of data that is sent when the UAV sends an authentication response message to the user. To reproduce an environment similar to a real UAV's storage, we used a data generator called *iozone* [21] to write dummy data to the SD card. For 3 minutes and 30 seconds, a total of 100 Mbytes of accumulated data was generated through the data generator. Figures 9(a) and 9(b) show the amount of data written to the SD card over the time interval. In Figure 9(a), the data generator always writes a constant amount of data, but in Figure 9(b), the amount of data to write largely changes with time. Figure 9(c) shows the amount of transferred data when the data is generated as shown in Figure 9(a), and Figure 9(d) shows the amount

of transferred data when the data is generated as shown in Figure 9(b). In both experiments, α was set to 2, 10, 50, and 100. Each experiment has two random selections of data, and the random selection of data occurred at the time indicated by t_{n-1} and t_n in Figure 9. For comparison, the random selection of data occurred at the same time in both experiments. Authentication was performed every 45 seconds.

In our system, the amount of transferred data after a random selection of data occurs is greatly increased. As can be seen from the experimental results, the increased amount of transferred data is highly dependent on α. The amount of generated data also affects the amount of data transferred, but the amount of data transferred is very small if α is small. In other words, our system makes it possible to select α depending on the situation, enabling timer-based erasure operation and erasure verification even when the maximum transmission amount of the UAV is largely restricted. In addition, even if the amount of generated data suddenly increases, a sample of generated data is selected and transmitted, so it is possible to communicate more stably by sending less data during unstable communication situations.

4.2. Erasure Operation Overhead Experiment. In this experiment, we measured the erasure latency according to the memory size of the UAV, where the erasure latency was the time from when the embedded timer reaches 0 to when the proof of erasure is generated. Our approach divides the memory region into three regions according to the type of data and performs the erasure operation. So, we divided the memory used in the experiment into the following regions: 5% of the total memory for the cold data region, 85% for the accumulated data region, and 10% for the hot data region. Since we wanted the erasure latency to be affected only by memory size in this experiment, we equally set all the variables except for memory size. In this experiment, the values of α and β were set to 2. Figure 10 shows each number of the proof transmit for 1 Gbytes, 2 Gbytes, 4 Gbytes, and 16 Gbytes memory size when a limited time of 8 minutes is given to the UAV that initiated the erasure operation. The data transfer rate via FSK is recommended to be 345 bits per second [22]. Therefore, our experiment also sent 345 bits per second when transferring the proof of erasure. The amount of data changed after communication was disconnected and was assumed to be 100 Mbytes. The erasure latency recorded in the graph is the average value of erasure latency measured through three times of the erase operation.

As we might expect, the erasure latency of first round increased with the memory size. In the rest of the experiment, except for the 1 Gbytes scenario, the difference between the erasure latency of the first round execution and the erasure latency of the second round execution was large. This is because the number of hash operations increased as memory size increased. The first transmission of proof of erasure takes 9 seconds longer than the subsequent proof transmission, since it contains the address value of the changed data

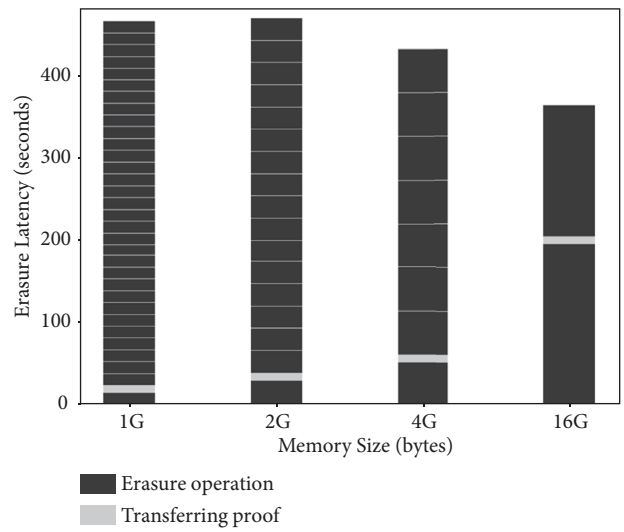

Figure 10: Experimental result for erasure latency according to the memory size of the unmanned aerial vehicle (UAV).

since communication was lost. As can be seen from the experimental results, if the UAV timer reaching zero, it has 8 minutes to perform the erasure operation 32 times at 1 Gbytes, 17 times at 2 Gbytes, 8 times at 4 Gbytes, and 2 times at 16 Gbytes. In other words, the UAV makes the best efforts in the given situation to erase the data and transfer the proof of erasure.

5. Conclusions

We have proposed a mechanism to provide proof of erasure operations after erasing data stored on UAVs, even if control of a remotely deployed UAV is lost. To do this, we used a countdown-based approach and hash chain to authenticate the sender of received messages and to trigger the erasure operation even after communication is lost. The accumulated data of the UAV is classified into new data types; the features of the data are actively utilized in the erasure operation, and the proof of erasure operation is transmitted so that the GCS can verify whether the erasure operation has actually been performed.

Our approach does not track all the data when generating proof of erasure, and so there is relatively little communication overhead; instead, we select seed data to generate proof of erasure. By allowing the amount of transferred data to generate proof through the value of α, timer-based erasure and erasure verification are possible even when the maximum amount of transmission is greatly limited. Furthermore, since a UAV that had lost control and started the erasure operation would not know what situation it was in, we used the best effort method to perform the erasure operation.

Data Availability

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

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) [Grant no. NRF-2017R1C1B2003957] and by the Institute for Information & Communications Technology Promotion (IITP) of the Korea government (MSIT) [Grant no. 2019-0-00426 and no. 2018-0-00420].

References

[1] D. Cenciotti, *Iran Unveils New UCAV Modeled on Captured U.S. RQ-170 Stealth Drone*, The Aviationist, 2016, https://theaviationist.com/2016/10/02/iran-unveils-new-ucav-modeled-on-captured-u-s-rq-170-stealth-drone/.

[2] S. Shane and D. E. Sanger, *Drone Crash in Iran Reveals Secret U.S. Surveillance Effort*, New York Times, 2011, https://www.nytimes.com/2011/12/08/world/middleeast/drone-crash-in-iran-reveals-secret-us-surveillance-bid.html/.

[3] L. A. White, "The need for governmental secrecy: why the us government must be able to withhold information in the interest of national security," *Virginia Journal of International Law*, vol. 43, p. 1071, 2002.

[4] N. Uchida, S. Takeuchi, T. Ishida, and Y. Shibata, "Mobile traffic accident prevention system based on chronological changes of wireless signals and sensors," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 8, no. 3, pp. 57–66, 2017.

[5] C. Gritti, M. Önen, R. Molva, W. Susilo, and T. Plantard, "Device identification and personal data attestation in networks," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 9, no. 4, pp. 1–25, 2018.

[6] N. Uchida, K. Ito, T. Ishida, and Y. Shibata, "Adaptive array antenna control methods with delay tolerant networking for the winter road surveillance system," *Journal of Internet Services and Information Security (JISIS)*, vol. 7, no. 1, pp. 2–13, 2017.

[7] F. Arena, P. Giovanni, and M. Collotta, "A survey on driverless vehicles: from their diffusion to security features," *Journal of Internet Services and Information Security (JISIS)*, vol. 8, no. 3, pp. 1–19, 2018.

[8] A. J. Kerns, D. P. Shepard, J. A. Bhatti, and T. E. Humphreys, "Unmanned aircraft capture and control via GPS spoofing," *Journal of Field Robotics*, vol. 31, no. 4, pp. 617–636, 2014.

[9] A. Rawnsley, "Iran's alleged drone hack: tough, but possible," *Wired*, 2011, https://www.wired.com/2011/12/iran-drone-hack-gps/.

[10] F. Hao, D. Clarke, and A. F. Zorzo, "Deleting secret data with public verifiability," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, pp. 617–629, 2016.

[11] M. D. Leom, K. Kwang, R. Choo, and R. Hunt, "Remote wiping and secure deletion on mobile devices: a review," *Journal of Forensic Sciences*, vol. 61, no. 6, pp. 1473–1492, 2016.

[12] X. Yu, Z. Wang, K. Sun, W. T. Zhu, N. Gao, and J. Jing, "Remotely wiping sensitive data on stolen smartphones," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA (CCS)*, pp. 537–542, ACM, Japan, 2014.

[13] S. Kim, T.-Y. Youn, D. Choi, and K.-W. Park, "Securely controllable and trustworthy remote erasure on embedded computing system for unmanned aerial vehicle," in *Proceedings of the 3rd International Symposium on Mobile Internet Security (MobiSec)*, vol. article 8, pp. 1–9, Cebu, Philippines, 2018.

[14] S. Kim, T.-Y. Youn, D. Choi, and K.-W. Park, "Securely controllable and trustworthy remote erasure on embedded computing system for unmanned aerial vehicle," in *Proceedings of the Research Briefs on Information & Communication Technology Evolution (ReBICTE)*, vol. 4, article 3, 2018.

[15] D. Perito and G. Tsudik, "Secure code update for embedded devices via proofs of secure erasure," in *Proceedings of the European Symposium on Research in Computer Security*, pp. 643–662, Springer, 2010.

[16] S. Dziembowski, T. Kazana, and D. Wichs, "One-time computable self-erasing functions," in *Theory of Cryptography*, vol. 6597 of *Lecture Notes in Computer Science*, pp. 125–143, Springer, Heidelberg, Germany, 2011.

[17] M. Ammar, B. Crispo, W. Daniels, and D. Hughes, "Speed: secure provable erasure for class-1 iot devices," in *Proceedings of the 8th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 111–118, 2018.

[18] G. O. Karame and W. Li, "Secure erasure and code update in legacy sensors," in *Proceedings of the International Conference on Trust and Trustworthy Computing*, vol. 9229, pp. 283–299, Springer International Publishing, 2015.

[19] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, 1981.

[20] M. Slonosky, "Trusted boot in COTS computing," *Military Embedded Systems*, 2015, http://mil-embedded.com/articles/trusted-boot-cots-computing/.

[21] W. D. Norcott, "Iozone filesystem benchmark," http://www.iozone.org/docs/IOzone_msword_98.pdf.

[22] L. Deshotels, "Inaudible sound as a covert channel in mobile devices," *WOOT*, 2014.