Marshall University

Marshall Digital Scholar

2020

DroneSig: Lightweight Digital Signature Protocol for Micro Aerial Vehicles

Yucheng Li

DRONESIG: LIGHTWEIGHT DIGITAL SIGNATURE PROTOCOL FOR MICRO AERIAL VEHICLES

A thesis submitted to
the Graduate College of
Marshall University
In partial fulfillment of
the requirements for the degree of
Master of Science
In
Computer Science
by
Yucheng Li
Approved by
Dr. Cong Pu, Committee Chairperson
Dr. Husnu Narman
Dr. Jamil Chaudri

Marshall University
August 2020

APPROVAL OF THESIS

We, the faculty supervising the work of Yucheng Li, affirm that the thesis, *DroneSig: Lightweight Digital Signature Protocol for Micro Aerial Vehicles*, meets the high academic standards for original scholarship and creative work established by the M.S. in Computer Science and the Department of Computer Sciences and Electrical Engineering. This work also conforms to the editorial standards of our discipline and the Graduate College of Marshall University. With our signatures, we approve the manuscript for publication.

Cong Pu

Dr. Cong Pu, Computer Sciences and Electrical Engineering   Committee Chairperson
Date
July 21, 2020

Husnu S. Narman

Dr. Husnu Narman, Computer Sciences and Electrical Engineering   Committee Member
Date
July 21, 2020

Jamil M Chaudri

Dr. Jamil Chaudri, Computer Sciences and Electrical Engineering   Committee Member
Date: *2020 Jul 22*

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Micro aerial vehicles a.k.a. drones, have become an integral part of a variety of civilian and military application domains, including but not limited to aerial surveying and mapping, aerial surveillance and security, aerial inspection of infrastructure, and aerial delivery. Meanwhile, the cybersecurity of drones is gaining significant attention due to both financial and strategic information and value involved in aerial applications. As a result of the lack of security features in the communication protocol, an adversary can easily interfere with on-going communications or even seize control of the drone. In this thesis, we propose a lightweight digital signature protocol, also referred to as DroneSig, to protect drones from a man-in-the-middle attack, where an adversary eavesdrops the communication between Ground Control Station (GCS) and drone, and impersonates the GCS and sends fake commands to terminate the on-going mission or even take control over the drone. The basic idea of the DroneSig is that the drone will only execute the new command after validating the received digital signature from the GCS, proving that the new command message is coming from the authenticated GCS. If the validation of the digital signature fails, the new command is rejected immediately, and the Return-to-Launch (RTL) mode is initiated and forces the drone to return to the take-off position. We conduct extensive simulation experiments for performance evaluation and comparison using OMNeT++, and simulation results show that the proposed lightweight digital signature protocol achieves better performance in terms of energy consumption and computation time compared to the standard Advanced Encryption Standard (AES) cryptographic technique.

I. INTRODUCTION

Micro aerial vehicles, a.k.a. drones, are flying robots endowed with the capabilities of sensing, computing, and wireless communicating, and becoming progressively popular in various civilian and military application areas, including but not limited to aerial surveying and mapping, aerial surveillance and security, aerial inspection of infrastructure, and aerial delivery (Pu & Carpenter, 2019). The global small drones market is projected to reach USD 40.31 billion by 2025, at a compound annual growth rate of 17.04% from 2018 to 2025 ("Global Small Drones Market," 2018). By 2026, commercial drones for both corporate and consumer applications will have an annual impact of $31 billion to $46 billion on the United States GDP ("Commercial drones are here," n.d.). As the drone-based civilian and military applications are proliferating, Internet of Drones (IoD), a layered aerial network management and control architecture, was proposed and has been demonstrated as an applicable architecture for coordinating the access of drones to controlled airspace and providing navigation services (Pu & Carpenter, 2020). With the assistance of advanced communication technology as well as emerging computing infrastructure, we envision that drones will definitely find many new ways to improve the quality of our life in the near future (Pu, 2019).

Due to both financial and strategic information and value involved in aerial applications, however, drones look especially attractive to attackers and become an ideal target for various cyber-attacks (Lin et al., 2018). For example, in January 2016, Mexican drug traffickers used satellite navigation signal deception technology to send spoofed GPS signals to attack the U.S. border patrol drone in order to illegally cross the border. In December 2011, Iran successfully captured a U.S. Lockheed Martin RQ-170 Sentinel drone through spoofing the drone's GPS system. Nowadays, drones have started showing their impact in everyday life of ordinary people

and have been considered as a supplement of humans in a part of the delivery in the business. Business and technology giants like Amazon, Google, Facebook, and Walmart have started delivering the products and services via drones for the speedy delivery and customer satisfaction. However, aerial drone applications are vulnerable to a myriad of cyber-attacks targeting their communication links with Ground Control Station (GCS), as well as with other air units (Sanjab et al., 2017). Therefore, investigating potential cybersecurity threats against drones and designing state-of-the-art security mechanisms are the top priority to improve the security of drone applications.

Unfortunately, the open nature of the wireless channel and the limited battery capacity, computing capability, and communication bandwidth make it become a highly challenging task (Pu, 2018). Communication between drones and GCS is established by the communication protocol via a wireless channel, which makes them vulnerable to various attacks since the communication protocol does not support security procedures (Koubaa et al., 2019). The GCS and drones exchange data through an unauthenticated wireless channel without encryption. Thus, data communication can be easily hacked. For example, an adversary can send unauthorized commands to the drone to take its control from GCS, and then catch and withhold the drone. This example is exactly showing that how the "anti-drone-gun" operates (*Dronebuster™*, 2018), or hijacking the drone to have it go to an arbitrary waypoint (Feng et al., 2019). Therefore, it is critical to ensure the security of communication in drone applications.

In this thesis, we propose a lightweight digital signature protocol, also named as DroneSig, to protect drones from man-in-the-middle (MITM) attack, where an adversary eavesdrops the communication between GCS and drone, and impersonates the GCS and sends fake commands to terminate the on-going mission or even take control over the drone. In the

DroneSig, the GCS generates a digital signature based on the command message by using the chaotic system and appends the digital signature to the command message. Before executing the received command, the drone validates the digital signature by comparing it to its own generated digital signature from the received command message. If the validation of the digital signature fails, the command is rejected immediately, and the Return-to-Launch (RTL) mode is initiated and forces the drone to return to take-off position. We develop a customized simulation framework and evaluate its performance through extensive simulations in terms of energy consumption, computation time, CPU cycle, memory usage, and code size. We also revisit prior AES, DES, and 3DES (Stallings, 2006), and modify them to work in the framework for performance comparison. The simulation results show that the proposed DroneSIG can achieve better performance in terms of energy consumption, computation time, CPU cycle, memory usage, and code size compared to AES, DES, and 3DES.

The rest of the thesis is organized as follows. Prior schemes are provided and analyzed in Section II. A system model and the proposed DroneSig are presented in Section III. Section IV focuses on simulation results and their analyses. Section V discusses the future work, Compare the security of the protocol. Finally, concluding remarks are provided in Section VI.

II. RELATED WORK

A significant volume of research work has mainly focused on developing security mechanisms and features to ensure the necessary security services of drones, such as confidentiality, integrity, and authentication, and protect drones from various cyber-attacks. Srinivas, Das, Kumar, and Rodrigues (2019) described a temporal credential-based anonymous lightweight user authentication mechanism is proposed to address the authentication problem in the IoD environment based on a three-factor scheme using user's mobile device, password, and

biometrics. Ozmen and Yavuz (2018) proposed an optimized public key infrastructure based framework integrated with lightweight symmetric primitives is proposed for small aerial drones, where special precomputation methods and optimized elliptic curves are harnessed to reduce the computational overhead and energy consumption. An encryption mechanism that improves the communication security of open source drones is proposed based on Galois Embedded Crypto (GEC) and ArduinoLibs Crypto library to provide safer and more secure communication service for radio control link (Podhradsky et al., 2017). A medium-interaction portable drone honeypot, also called HoneyDrone, is designed for protecting drones by Daubert, Boopalan, Mühlhäuser, and Vasilomanolakis (2018). The basic idea of HoneyDrone is to emulate a number of drone-specific and drone-tailored protocols, lure adversary into attacking drone honeypot, and record and analyze malicious activities to detect potential attackers.

According to Won, Seo and Bertino (2020), a lookup table shuffling mechanism that supports white-box block cipher with dynamics is proposed to protect unmanned vehicles from white-box attacks, where attackers with sufficient knowledge of a target unmanned vehicle can steal secret information stored in the unmanned vehicle through taking advantage of advanced reverse engineering techniques and exploiting the vulnerabilities of open-source software. Since no short secret key is used by an unmanned vehicle during the protocol, the shuffling mechanism can be safely executed in the white-box environment and make it hard for a white-box attacker to successfully encrypt/decrypt any plaintext/ciphertext even if the attacker has the knowledge of the entire lookup table. A new system model is proposed to secure drone communication for the data collection and transmission in the IoD environment, where public blockchain technology is used for the storage of collected data from the drones and update the information into the distributed ledgers to reduce the burden of drones (Aggarwal et al., 2019). According to

experimental evaluation, the proposed system model makes the realtime drone-based applications more reliable and scalable and can defend against various risks and attacks.

He, Qiao, Chan, and Guizani (2018) proposed that use information fusion by combining a visual sensor and inertial measurement unit to detect GPS spoofing attacks in an airborne fog computing system. In order to address the challenging information leakage problem of eavesdropping attack, leverages the physical characteristics of wireless channels to achieve the goal of secure transmissions in unmanned aerial vehicles communication networks (Li et al., 2019). In addition, an overview of security threats and attacks against communication protocol for unmanned systems and potential security solutions are also presented by Koubaa et al. (2019). Liang, Zhao, Shetty, and Li (2017) proposed a blockchain and cloud storage-based framework to guarantee the UAV data integrity. The hashed data records collected from drones are stored in the blockchain network, and a blockchain receipt for each data record is also stored in the cloud, which can reduce the burden of moving drones with the limit of battery and process capability while gaining enhanced security guarantee of the data. The article presents the ideology of the secure utilization of drones for inter-service operability in ultra-dense wireless networks by exploiting the features of the blockchain (Sharma et al., 2017). Zhang, He, Li, and Chen (2020) proposed a lightweight authentication and key agreement scheme in which there are only secure one-way hash function and bitwise XOR operations when drones and users mutually authenticate each other. The proposed scheme is comprised of three phases: the setup phase, the registration phase, and the mutual authentication phase. In the setup phase, the control station generates its master private key and other public system parameters. In the registration phase, users and drones register on the control station and get their secret key via a secure channel. In

the last phase, users and drones communicate with each other securely after establishing a session key.

In summary, various cryptographic techniques have been well studied to protect drones from cyber-attacks. However, to the best of our knowledge, there is no comprehensive and lightweight defense mechanism against MITM attack for drones.

III. THE PROPOSED LIGHTWEIGHT DIGITAL SIGNATURE PROTOCOL

In this section, we first introduce the system model and chaotic system, then propose a lightweight digital signature protocol, also named as DroneSig, to protect drones from man-in-the-middle (MITM) attack.
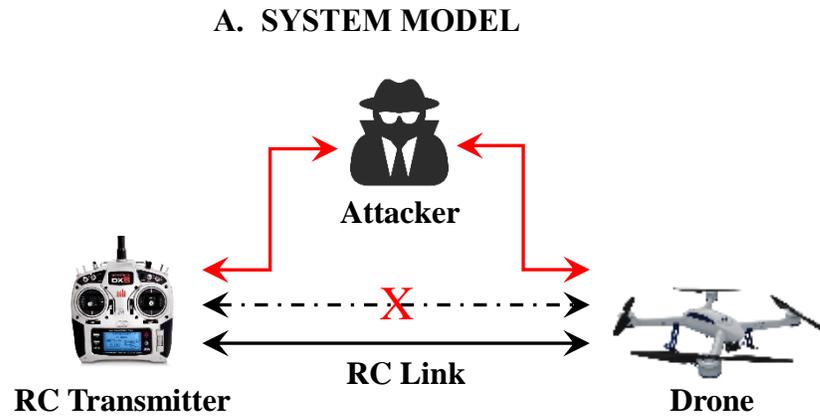
A. SYSTEM MODEL

Figure 1. System model.

This image shows a basic system diagram where there is a Radio Control (RC) link to be used by the GCS to manually control the drone. However, the communication link between GCS and drone is established via a wireless channel, which is vulnerable to various security attacks due to its openness. To be specific, the GCS exchanges data with the drone through an unauthenticated and unencrypted channel; as a result, the communications can be easily hacked by a man-in-the-middle (MITM) attack. An adversary with an appropriate RC transmitter can

eavesdrop the communication between GCS and drone and impersonates the GCS and sends

fake commands to terminate the on-going mission or even gain direct control over the drone

(Srinivas et al., 2019). Here, a successful communication link attack without involving "anti-

drone-gun" has already been demonstrated on a popular DSMx radio protocol to hijack the drone

(Liang et al., 2017).

B. CHAOTIC SYSTEM

A chaotic system is a dynamical and determined system with the extrinsic nature of

nonlinear behavior, pseudo-randomness, broad-spectrum, and sensitivity to initial conditions. In

the past few decades, a state of disorder and nonlinear dynamics have been used in the design of

cryptographically secure pseudo-random number generators. These pseudo-random number

generators use the control parameters and the initial condition of the chaotic maps as their keys.

Without the right initial conditions, the correct pseudo-random sequence cannot be regenerated.

Duffing map is a two-dimensional discrete-time and dynamical system that exhibits chaotic

behavior. It is widely known to display chaos for certain parameter values and initial conditions.

Duffing map contains a single cubic term and is expressed below,

$$\begin{cases} x_{n+1} = y_n \\ y_{n+1} = -b \cdot x_n + a \cdot y_n - y_n^3 \end{cases}$$

Table 1. Duffing Map

where a and b are constant parameters. The output of the Duffing map highly depends on

the initial conditions represented by x_0 and y_0. The constant parameters are usually sent to a =

2.75 and b = 0.2 to produce chaotic behavior. Disregarding the initial point (0.5, 0.5), the Duffing

map outputs points around the Duffing map attractor in a random way.
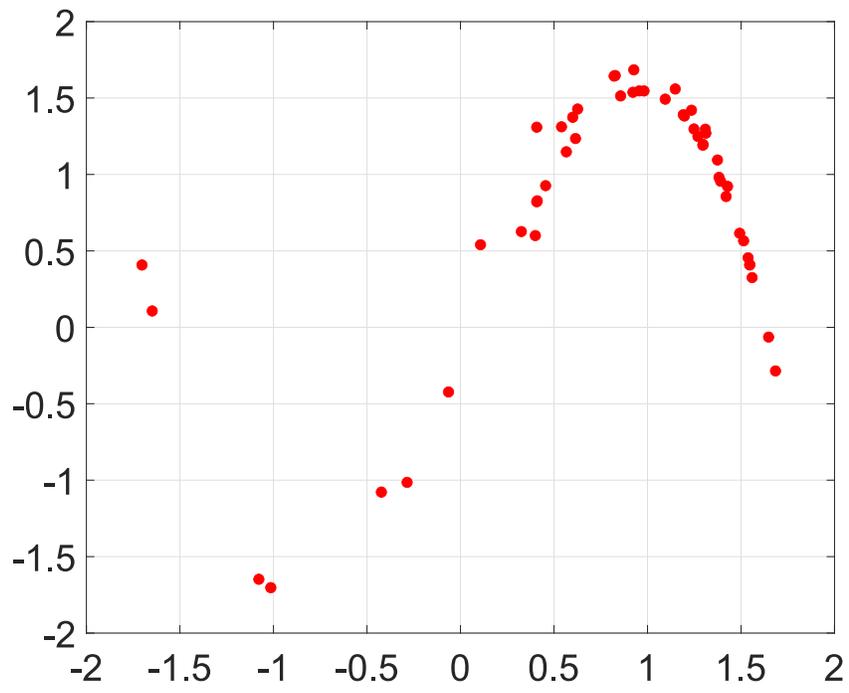
(a) $x_0 = 0.4$ and $y_0 = 0.6$
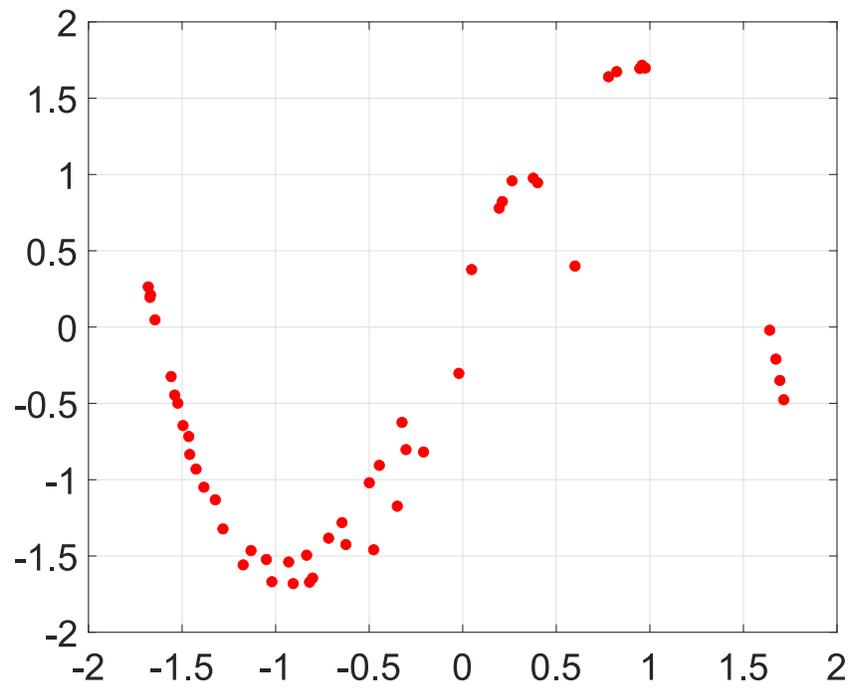
(b) $x_0 = 0.6$ and $y_0 = 0.4$
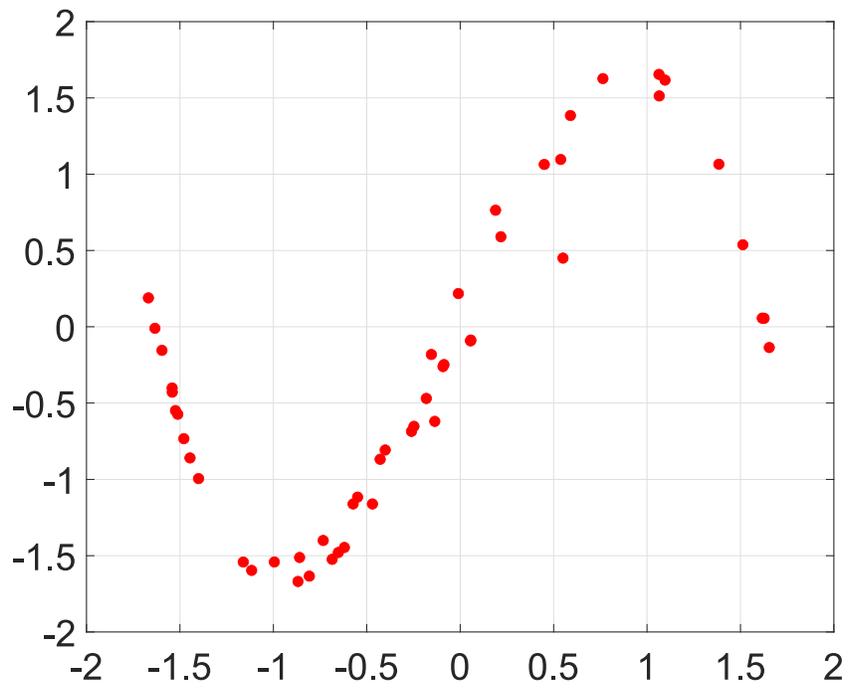
(c) $x_0 = 0.55$ and $y_0 = 0.45$
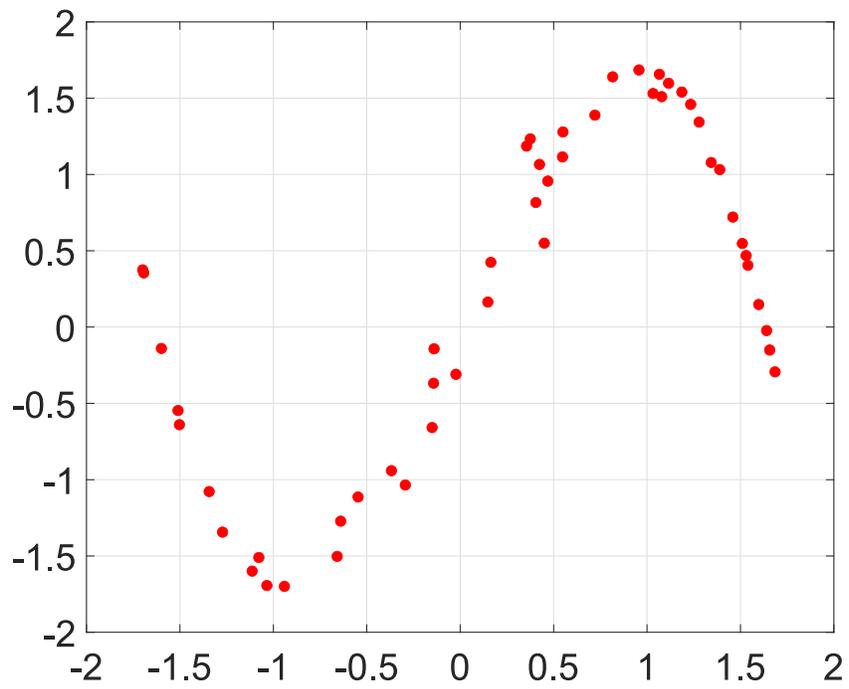
(d) $x_0 = 0.45$ and $y_0 = 0.55$

Figure 2. Duffing map with different initial conditions after 50 iterations.

As shown in Figure 2, any change in the initial conditions will affect the plot of these points.

C. LIGHTWEIGHT DIGITAL SIGNATURE PROTOCOL

The DroneSig adopts a technique that is similar to cryptographic encryption but requires less computational resources. In addition, the DroneSig is designed to encode and decode binary information without using standard cryptographic techniques, such as DES or AES. In DroneSig, the digital signature is generated by using a random number generator, Duffing map, which can assist both GCS and drone to achieve the same key without the necessity to wirelessly share it on a public wireless medium.

The DroneSig consists of three functions: byte substitution, matrix transformation, and random shuffling.

Message Command (256 Bytes)

Byte Substitution

| m_1 | m_2 | m_3 | m_4 | m_5 | m_6 | m_7 | m_8 | m_9 | m_{10} | m_{11} | m_{12} | m_{13} | m_{14} | m_{15} | m_{16} |

Inverse in $GF(2^8)$

$$\begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \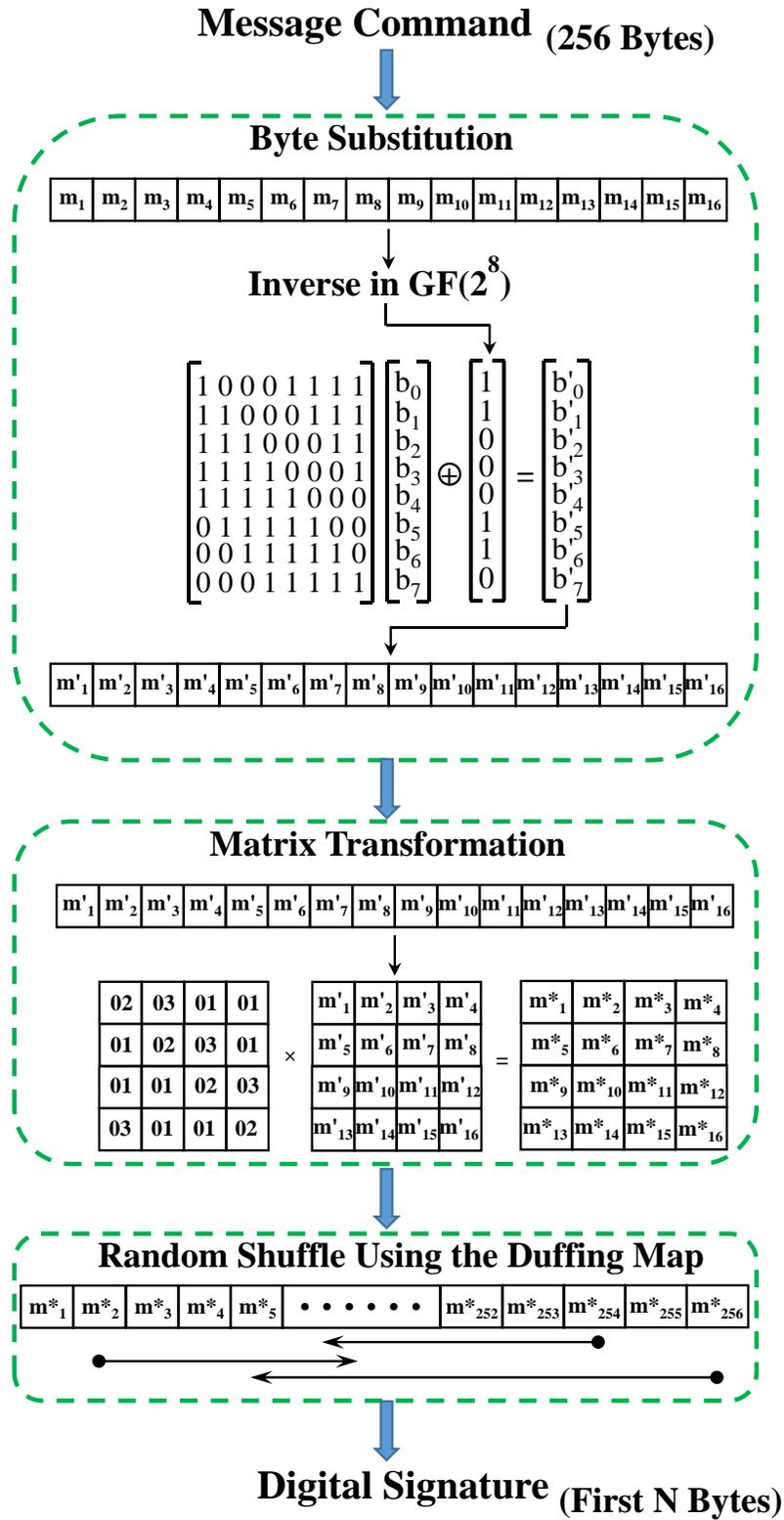\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix}$$

| m'_1 | m'_2 | m'_3 | m'_4 | m'_5 | m'_6 | m'_7 | m'_8 | m'_9 | m'_{10} | m'_{11} | m'_{12} | m'_{13} | m'_{14} | m'_{15} | m'_{16} |

Matrix Transformation

| m'_1 | m'_2 | m'_3 | m'_4 | m'_5 | m'_6 | m'_7 | m'_8 | m'_9 | m'_{10} | m'_{11} | m'_{12} | m'_{13} | m'_{14} | m'_{15} | m'_{16} |

$$\begin{bmatrix} 02&03&01&01 \\ 01&02&03&01 \\ 01&01&02&03 \\ 03&01&01&02 \end{bmatrix} \times \begin{bmatrix} m'_1&m'_2&m'_3&m'_4 \\ m'_5&m'_6&m'_7&m'_8 \\ m'_9&m'_{10}&m'_{11}&m'_{12} \\ m'_{13}&m'_{14}&m'_{15}&m'_{16} \end{bmatrix} = \begin{bmatrix} m^*_1&m^*_2&m^*_3&m^*_4 \\ m^*_5&m^*_6&m^*_7&m^*_8 \\ m^*_9&m^*_{10}&m^*_{11}&m^*_{12} \\ m^*_{13}&m^*_{14}&m^*_{15}&m^*_{16} \end{bmatrix}$$

Random Shuffle Using the Duffing Map

| m^*_1 | m^*_2 | m^*_3 | m^*_4 | m^*_5 | • • • • • • • | m^*_{252} | m^*_{253} | m^*_{254} | m^*_{255} | m^*_{256} |

Digital Signature (First N Bytes)

Figure 3. The overall structure of the DroneSig.

Figure 3 shows the overall structure of the DroneSig. Each message command has 256 bytes and is divided into a set of 16-byte blocks. Byte substitution and matrix transformation will be applied to each block, whilst random shuffling will be performed on all blocks. First, each individual byte of 16-byte block is mapped into a new byte according to

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
b_0^* \\ b_1^* \\ b_2^* \\ b_3^* \\ b_4^* \\ b_5^* \\ b_6^* \\ b_7^*
\end{bmatrix}
\oplus
\begin{bmatrix}
1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0
\end{bmatrix}
=
\begin{bmatrix}
b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7'
\end{bmatrix}
$$

Table 2. Byte Substitution (1)

Here, $(b_7^* b_6^* b_5^* b_4^* b_3^* b_2^* b_1^* b_0^*)$ is the value of multiplicative inverse in GF (2^8) for input byte $(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$. As an example, considering the input byte {95}. the multiplicative inverse in GF (2^8) is $\{95\}^{-1} = \{8A\}$, which is (10001010).

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1
\end{bmatrix}
\oplus
\begin{bmatrix}
1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0
\end{bmatrix}
$$

Table 3. Byte Substitution (2)

According to Table 3, the result byte is {2A}. The process of byte substitution is done.

Second, the 16 substituted bytes in a block are depicted as a 4×4 square matrix, and each byte of a column in square matrix is mapped into a new value that is a function of all four bytes

in that column. The transformation is defined by matrix transformation in Figure 3. Each element

in the product matrix is the sum of products of elements of one row and one column. In this case,

the individual additions and multiplications are performed in GF (2^8). The matrix transformation

on a single column can be expressed as

$$S'_{0,j} = (2 \cdot S_{0,j}) \oplus (3 \cdot S_{1,j}) \oplus S_{2,j} \oplus S_{3,j}$$

$$S'_{1,j} = S_{0,j} \oplus (2 \cdot S_{1,j}) \oplus (3 \cdot S_{2,j}) \oplus S_{3,j}$$

$$S'_{2,j} = S_{0,j} \oplus S_{1,j} \oplus (2 \cdot S_{2,j}) \oplus (3 \cdot S_{3,j})$$

$$S'_{3,j} = (3 \cdot S_{0,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus (2 \cdot S_{3,j})$$

Table 4. Mix Columns

Third, the 256 bytes of all blocks will be randomly shuffled using the Duffing map to

generate the digital signature, which includes the first N bytes of the shuffling output. The

shuffling process is reversible. When the drone receives the command message, it only executes

the command after verifying the authenticity of the digital signature, proving that the

communication has been held with the authenticated GCS. The drone will validate the digital

signature by comparing it to its own generated signature from the command message. If this

validation of the digital signature fails, the command is rejected immediately, and the Return-to-

Launch (RTL) mode is initiated and forces the drone to return to take-off position.

IV. PERFORMANCE EVALUATION

In this thesis, we develop a customized simulation framework to conduct our experiments

in terms of code size, memory usage, energy consumption, computation time, and CPU cycle.

We also revisit existing AES, DES, and 3DES (Stallings, 2006), and modify them to work in the

framework for performance comparison and analysis. The size of plaintext is changed between

25 and 2000 KB.

(a)

(c)
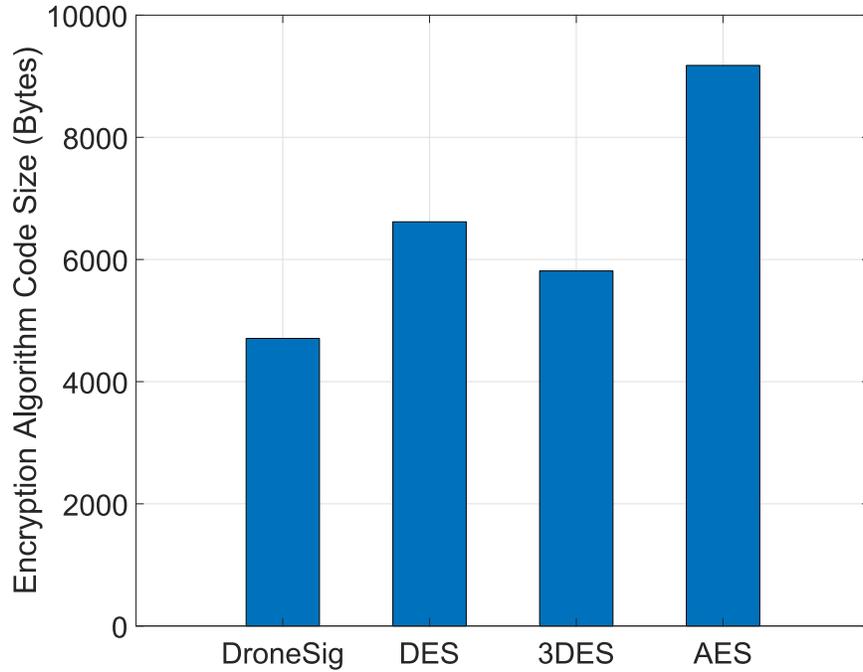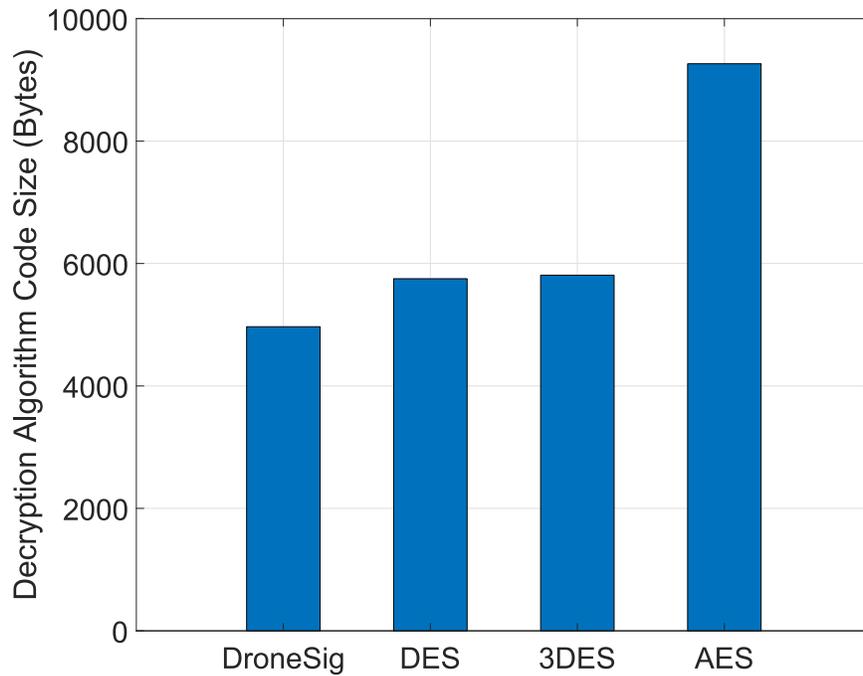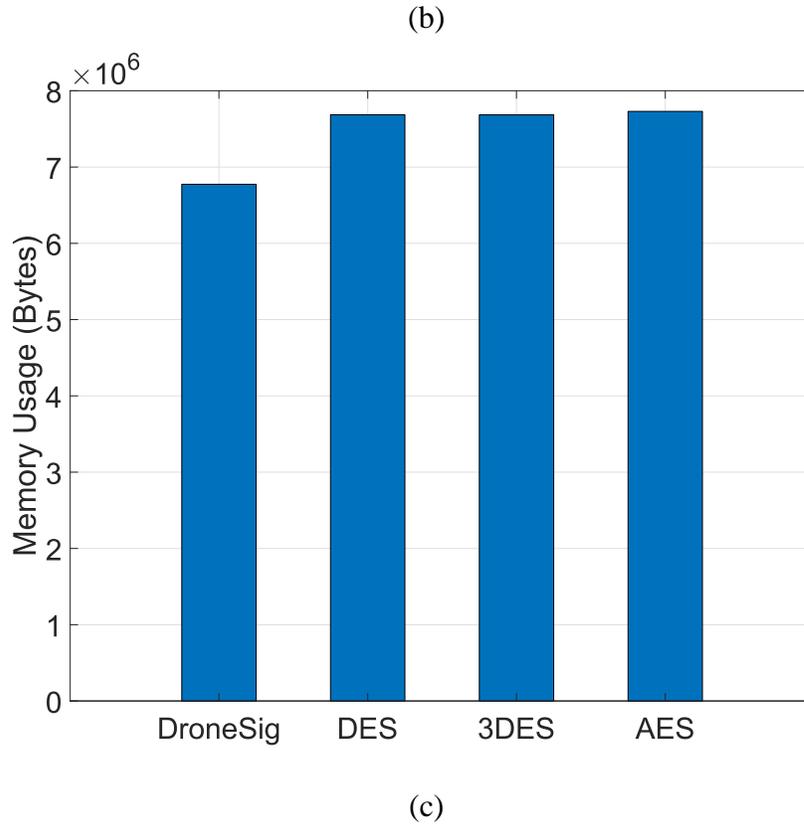
Figure 4. Performance of code size and memory usage against the size of plaintext.

First, the performance of code size and memory usage is measured with the changes in the size of plaintext in Figure 4. Here, the code size is measured as the file size of the algorithm. As shown in Figure 4(a) and (b), the DroneSig has the smallest code size in terms of encryption and decryption algorithms compared to AES, DES, and 3DES. Because the DroneSig has a less number of operations for encryption and decryption processes, which make the file size of algorithms smaller. The AES has the largest code size in terms of encryption and decryption algorithms because it is the most complex algorithm which consists of four transformation functions: substitute bytes, shift rows, mix columns, and add round key. In Figure 4(c), we measure the memory usage of four schemes. It is clear that the DroneSig has the smallest memory usage compared to AES, DES, and 3DES.

(a)

(b)

16
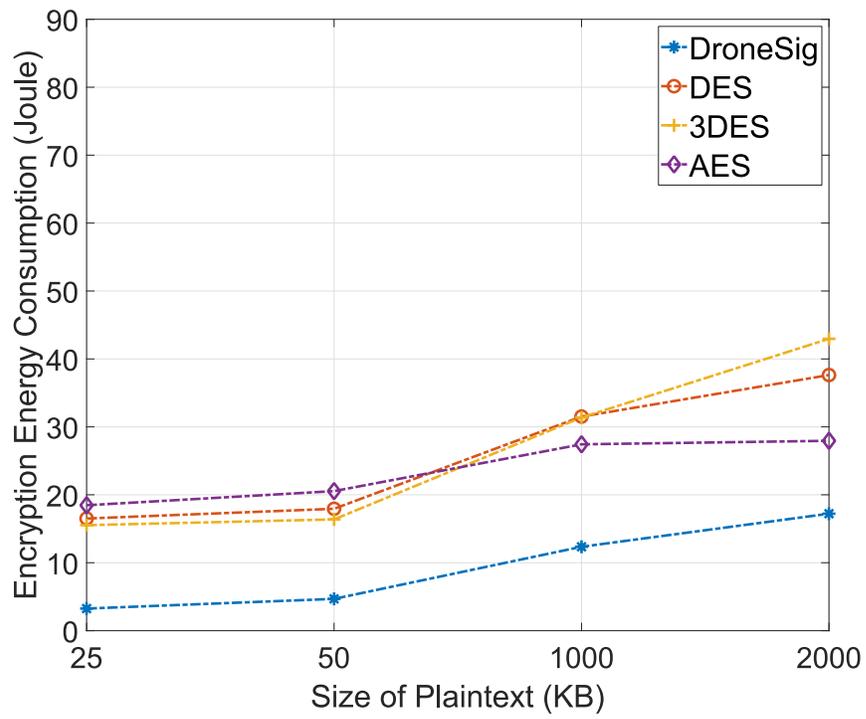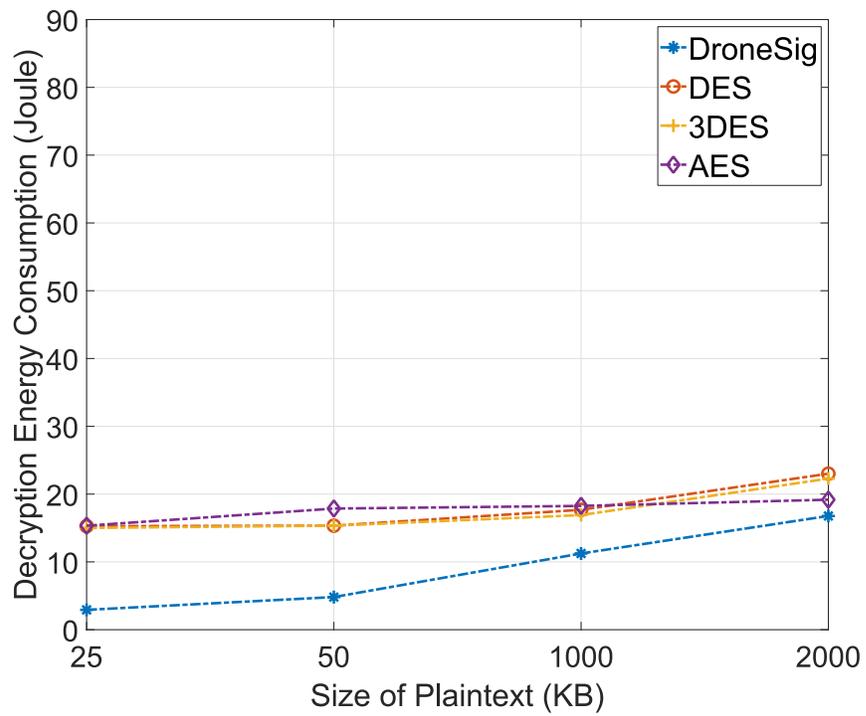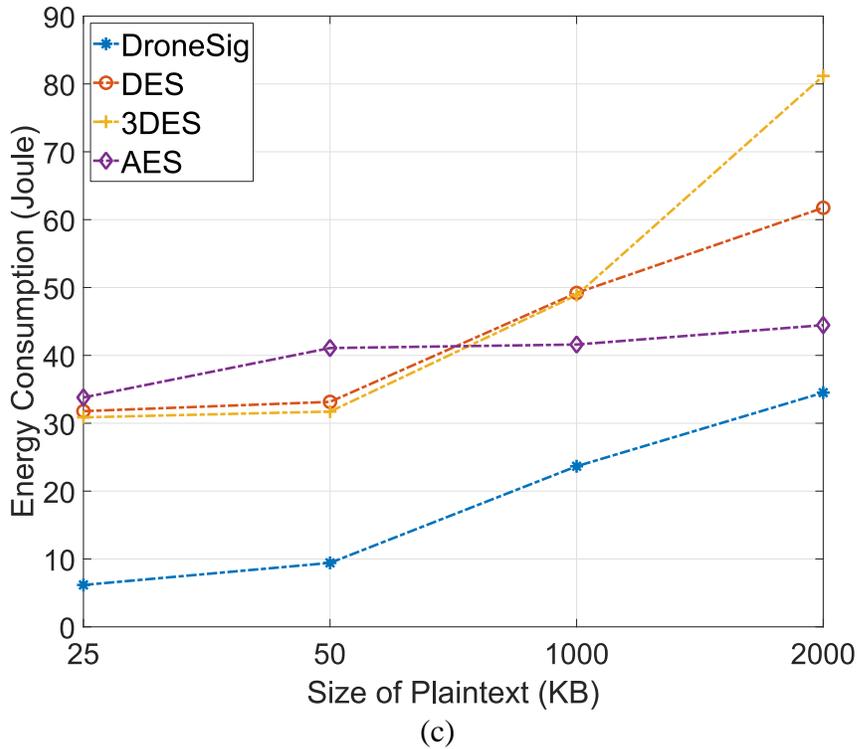
Figure 5. Performance of energy consumption against the size of plaintext.

Second, we measure the performance of energy consumption against the size of plaintext in Figure 5. As shown in Figure 5(a), the DroneSig achieves the lowest encryption energy consumption compared to AES, DES, and 3DES. DroneSig consumes less energy because it performs three lightweight operations: byte Substitution, matrix Transformation, and random Shuffling. Most importantly, three lightweight operations are only executed one time in the process of encryption. Thus, the lowest encryption energy consumption is observed by the DroneSig. However, for AES, DES, and 3DES, the same encryption operations are performed in multiple rounds. As a result, a large amount of energy is consumed. In Figure 5(b), it is clear that the decryption energy consumption of the DroneSig is lower than that of the other three schemes. Since the decryption is the reverse process of encryption, similar operations will be applied to ciphertext. Therefore, the lowest decryption energy consumption is obtained by the DroneSig.

17

The total energy consumption is measured in Figure 5(c), where the DroneSig provides the lowest total energy consumption compared to AES, DES, and 3DES. Because the DroneSig has the lowest encryption and decryption energy consumption.

(a)

18
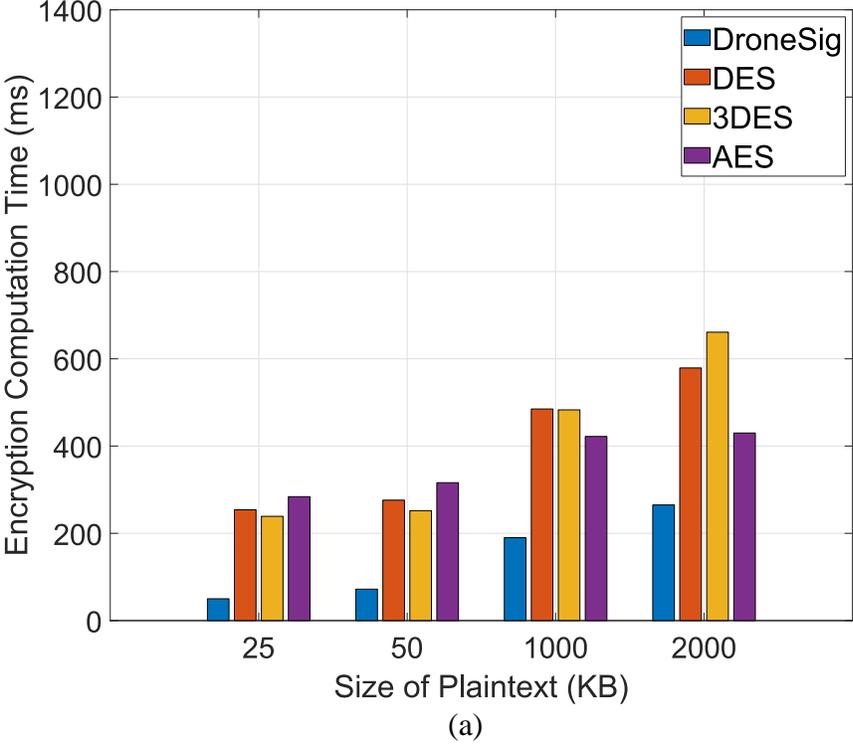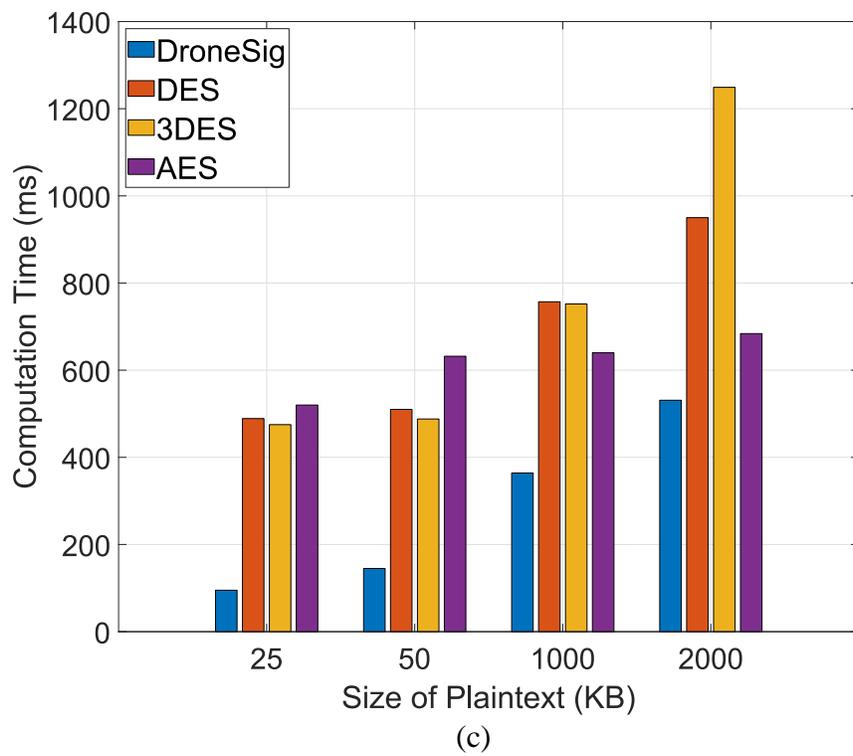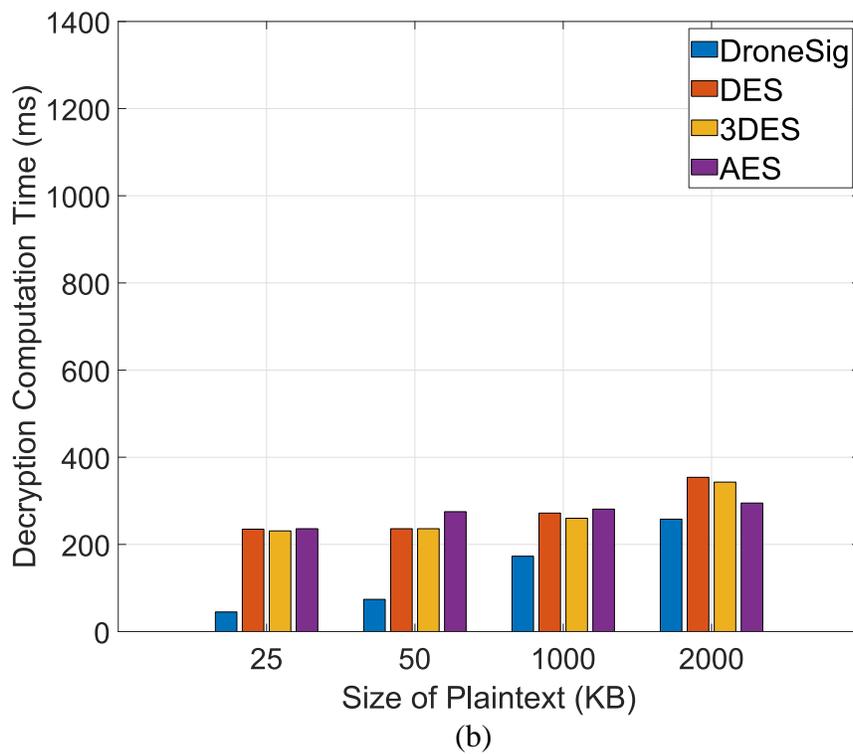
Figure 6. Performance of computation time against the size of plaintext.

Third, the performance of computation time is measured with varying size of plaintext in Figure 6. The computation time is proportional to the complexity of the algorithm. As the algorithm becomes more complex, it requires a larger computation time. In the DroneSig, there are only three operations, and those operations are only executed one time for encryption and decryption. However, AES, DES, and 3DES are traditional cryptographic techniques, and several complex operations are executed in multiple rounds for encryption and decryption. Compared to DroneSig, AES, DES, and 3DES are much more complex. Therefore, the DroneSig can achieve the shortest computation time in terms of encryption and decryption, which are shown in Figure 6(a) and (b), respectively. In Figure 6(c), the DroneSig outperforms AES, DES, and 3DES in terms of total computation time because the DroneSig can achieve the smallest encryption and decryption computation time.

(a)
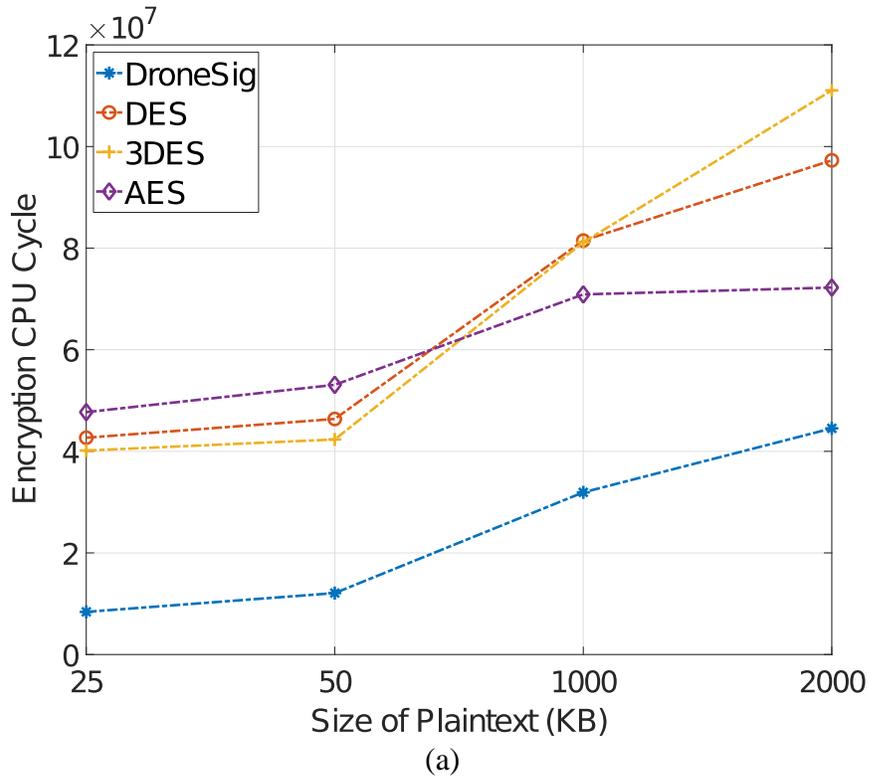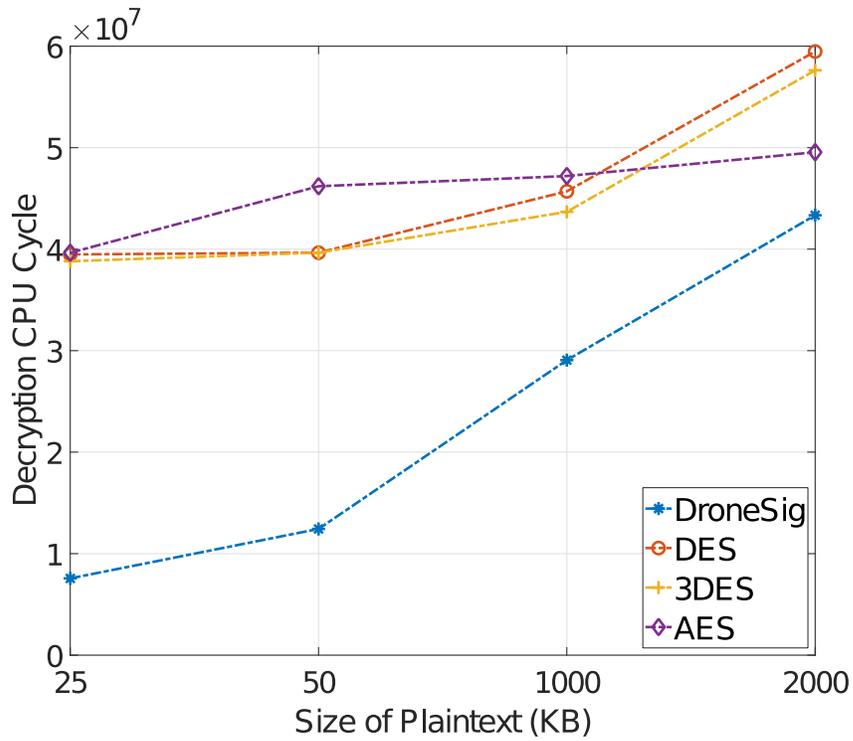
(b)
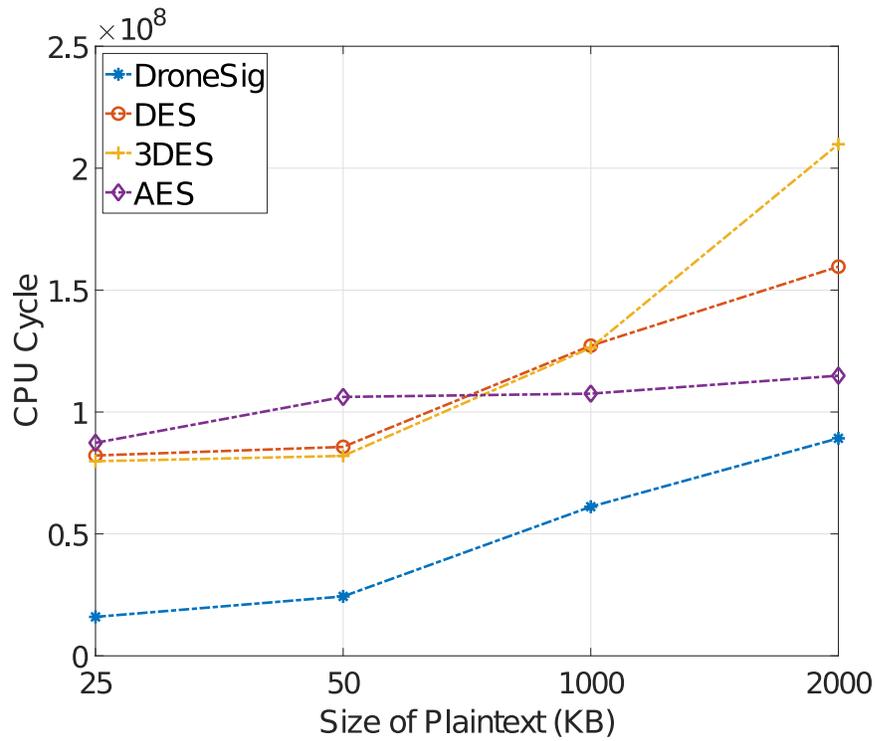
(c)

Figure 7. Performance of CPU cycle against the size of plaintext.

21

Fourth, we measure the performance of the CPU cycle by changing the size of plaintext in Figure 7. As shown in Figure 7(a) and (b), the smallest number of CPU cycles is obtained by the DroneSig in terms of encryption and decryption. Since the DroneSig significantly reduces the number of operations in the process of encryption and decryption, a smaller number of CPU cycles is required to complete the operations of encryption and decryption. However, AES, DES, and 3DES are more complex than DroneSig. Thus, a larger number of CPU cycles is required to execute all operations. In Figure 7, the total number of encryption and decryption CPU cycles is measured for all schemes. The DroneSig provides the best performance compared to others because the DroneSig can achieve a smaller number of CPU cycles in terms of encryption and decryption.

V. FUTURE WORK

Although the analysis and simulation provided have proved that DroneSig is a lightweight, low-consumption, and fast calculation method, there are some improvements that can still be made. In Section IV, we compared code size, memory usage, energy consumption, computation time, and CPU cycle. Nevertheless, these seem to only show that the performance of DroneSig is good, but they cannot explain how safe it is. In this context, we will survey some of the provided results, which can be improved or extended further.

Therefore, after a series of studies, we found two lightweight Micro Aerial Vehicle protocols to compare with DroneSig. The first one is Micro Air Vehicle Communication Protocol (MAVLink), a very lightweight, header-only message library for communication between drones and/or ground control stations (Meier, 2009). The second is UranusLink, Communication Protocol for UAV with Small Overhead and Encryption Ability (Kriz & Gabrlik, 2015).

We will deploy two attack methods at the same time to compare the security of each protocol. One type of attack can be an attempt to send random data in the encrypted portion and attempt to provide some combination of valid commands on the receiver after decryption, see if the length of the received data has changed. Another type of attack can be capturing the communication during flight and try to use the captured comment on the next flight to see if the drone executes.

To determine the security of cryptographic protocols, we intend to use the ISO/IEC 29128 method for evaluation testing. ISO/IEC 29128 is a newly proposed international standard that USES formal methods, which used for improving the security assurance of cryptographic protocols (Matsuo et al., 2010). Once the cryptographic protocol has passed ISO/IEC 29128 certification, especially at its highest level of assurance, the protocol will be absolutely secure.

VI. CONCLUSION

In this thesis, we proposed a lightweight digital signature protocol (DroneSig) to protect drones from a man-in-the-middle attack, where an adversary eavesdrops the communications between Ground Control Station and drone, and impersonates the Ground Control Station and sends fake commands to terminate the on-going mission or even take control over the drone. The basic idea of the DroneSig is that the drone will only execute the new command after validating the received digital signature from the Ground Control Station, proving that the new command message is coming from the authenticated Ground Control Station. If the validation of the digital signature fails, the new command is rejected immediately, and the Return-to-Launch (RTL) mode is initiated and forces the drone to return to the take-off position. In order to evaluate the effectiveness of the proposed approach, we developed a customized simulation framework and compared it with prior approaches. The simulation results show that the proposed DroneSig is a

viable and competitive approach defending drones against a man-in-the-middle attack.

REFERENCES

Aggarwal, S., Shojafar, M., Kumar, N., & Conti, M. (2019). A New Secure Data Dissemination

    Model in Internet of Drones. *2019 IEEE International Conference on*

    Communications (ICC), 1–6. https://doi.org/10.1109/icc.2019.8761372

Commercial drones are here: The future of unmanned aerial systems. McKinsey & Company.

    https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-

    insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems.

Daubert, J., Boopalan, D., Mühlhäuser, M., & Vasilomanolakis, E. (2018). Improving

    communication security of open source UAVs: Encrypting radio control link. *2018*

    Network Operations and Management Symposium (NOMS), 1-6.

Dronebuster™. Flex Force. (2018, March 29). http://flexforce.us/product/dronebuster/.

Feng, Z., Guan, N., Lv, M., Liu, W., Deng, Q., Liu, X., & Yi, W. (2019). An Efficient UAV

    Hijacking Detection Method Using Onboard Inertial Measurement Unit. *ACM*

    Transactions on Embedded Computing Systems, *17*(6), 1–19.

    https://doi.org/10.1145/3289390

Global Small Drones Market 2018-2025 - Exemptions Made By the FAA to Allow the Use of

    Small Drones in Several Industries. PR Newswire: news distribution, targeting and

    monitoring. (2018, October 1). https://www.prnewswire.com/news-releases/global-small-

    drones-market-2018-2025---exemptions-made-by-the-faa-to-allow-the-use-of-small-

    drones-in-several-industries-300721794.html.

Goodin, D. (2016, October 27). *There's a new way to take down drones, and it doesn't involve*

    shotguns. Ars Technica. https://arstechnica.com/information-technology/2016/10/drone-

    hijacker-gives-hackers-complete-control-of-aircraft-in-midflight/.

He, D., Qiao, Y., Chan, S., & Guizani, N. (2018). Flight Security and Safety of Drones in

Airborne Fog Computing Systems. *IEEE Communications Magazine*, *56*(5), 66–71.

https://doi.org/10.1109/mcom.2018.1700916

Koubaa, A., Allouch, A., Alajlan, M., Javed, Y., Belghith, A., & Khalgui, M. (2019). Micro Air

Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access*, *7*, 87658–87680.

https://doi.org/10.1109/access.2019.2924410

Kriz, V., & Gabrlik, P. (2015). UranusLink - Communication Protocol for UAV with Small

Overhead and Encryption Ability. *IFAC-PapersOnLine, 48*(4), 474-479.

https://doi:10.1016/j.ifacol.2015.07.080

Li, B., Fei, Z., Zhang, Y., & Guizani, M. (2019). Secure UAV Communication Networks over

5G. *IEEE Wireless Communications*, *26*(5), 114–120.

https://doi.org/10.1109/mwc.2019.1800458

Liang, X., Zhao, J., Shetty, S., & Li, D. (2017). Towards data assurance and resilience in IoT

using blockchain. *2017 IEEE Military Communications Conference (MILCOM)*, 261–

266. https://doi.org/10.1109/milcom.2017.8170858

Lin, C., He, D., Kumar, N., Choo, K.-K. R., Vinel, A., & Huang, X. (2018). Security and Privacy

for the Internet of Drones: Challenges and Solutions. *IEEE Communications Magazine*,

56(1), 64–69. https://doi.org/10.1109/mcom.2017.1700390

Matsuo, S., Miyazaki, K., Otsuka, A., & Basin, D. (2010). How to evaluate the security of real-

life cryptographic protocols. In *International Conference on Financial Cryptography

and Data Security* (pp. 182-194). Springer, Berlin, Heidelberg.

Meier, L. (2009). *MAVLink Developer Guide*. Retrieve from https://mavlink.io/en/

Ozmen, M. O., & Yavuz, A. A. (2018). Dronecrypt - An Efficient Cryptographic Framework for Small Aerial Drones. *2018 IEEE Military Communications Conference (MILCOM)*, 1–6. https://doi.org/10.1109/milcom.2018.8599784

Podhradsky, M., Coopmans, C., & Hoffer, N. (2017). Improving communication security of open source UAVs: Encrypting radio control link. *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1153–1159. https://doi.org/10.1109/icuas.2017.7991460

Pu, C. (2018). Jamming-Resilient Multipath Routing Protocol for Flying Ad Hoc Networks. *IEEE Access*, *6*, 68472–68486. https://doi.org/10.1109/access.2018.2879758

Pu, C. (2019). Stochastic Packet Forwarding Algorithm in Flying Ad Hoc Networks. *2019 IEEE Military Communications Conference (MILCOM)*, 490–495. https://doi.org/10.1109/milcom47813.2019.9020723

Pu, C., & Carpenter, L. (2019). To Route or To Ferry: A Hybrid Packet Forwarding Algorithm in Flying Ad Hoc Networks. *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 1–8. https://doi.org/10.1109/nca.2019.8935011

Pu, C., & Carpenter, L. (2020). Psched: A Priority-Based Service Scheduling Scheme for the Internet of Drones. *IEEE Systems Journal (Early Access)*, 1–1. https://doi.org/10.1109/jsyst.2020.2998010

Pu, C., & Li, Y. (2020). Lightweight Authentication Protocol for Unmanned Aerial Vehicles Using Physical Unclonable Function and Chaotic System. *2020 IEEE International Symposium on Local and Metropolitan Area Networks (*LANMAN*), Accepted to Appear.*

Sanjab, A., Saad, W., & Basar, T. (2017). Prospect theory for enhanced cyber-physical security

    of drone delivery systems: A network interdiction game. *2017 IEEE International*

    Conference on Communications (ICC), 1–6. https://doi.org/10.1109/icc.2017.7996862

Wang, B., Sun, Y., Sheng, Z., Nguyen, H. M., & Duong, T. Q. (2019). Inconspicuous

    Manipulation for Social-Aware Relay Selection in Flying Internet of Things. *IEEE*

    Wireless Communications Letters, *8*(5), 1394–1397.

    https://doi.org/10.1109/lwc.2019.2919536

Sharma, V., You, I., & Kul, G. (2017). Socializing Drones for Inter-Service Operability in Ultra-

    Dense Wireless Networks using Blockchain. *Proceedings of the 2017 International*

    Workshop on Managing Insider Security Threats - MIST '17, 81–84.

    https://doi.org/10.1145/3139923.3139932

Srinivas, J., Das, A. K., Kumar, N., & Rodrigues, J. J. P. C. (2019). TCALAS: Temporal

    Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones

    Environment. *IEEE Transactions on Vehicular Technology*, *68*(7), 6903–6916.

    https://doi.org/10.1109/tvt.2019.2911672

Stallings, W. (2006). *Cryptography and network security: principles and practice*. Pearson.

Won, J., Seo, S., & Bertino, E. (2020). A Secure Shuffling Mechanism for White-Box Attack-

    Resistant Unmanned Vehicles. *IEEE Transactions on Mobile Computing*, *19*(5), 1023–

    1039. https://doi.org/10.1109/tmc.2019.2903048

Zhang, Y., He, D., Li, L., & Chen, B. (2020). A lightweight authentication and key agreement

    scheme for Internet of Drones. *Computer Communications*, *154*, 455–464.

    https://doi.org/10.1016/j.comcom.2020.02.067

APPENDIX A: APPROVAL LETTER

MARSHALL UNIVERSITY.
www.marshall.edu

Office of Research Integrity

May 4, 2020

Yucheng Li
620 15th Street, Apt 3
Huntington, WV 25701

Dear Yucheng:

This letter is in response to the submitted thesis abstract entitled *"DroneSig: Lightweight Digital Signature Protocol for Micro Aerial Vehicles."* After assessing the abstract, it has been deemed not to be human subject research and therefore exempt from oversight of the Marshall University Institutional Review Board (IRB). The Code of Federal Regulations (45CFR46) has set forth the criteria utilized in making this determination. Since the information in this study does not involve human subjects as defined in the above referenced instruction, it is not considered human subject research. If there are any changes to the abstract you provided then you would need to resubmit that information to the Office of Research Integrity for review and a determination.

I appreciate your willingness to submit the abstract for determination. Please feel free to contact the Office of Research Integrity if you have any questions regarding future protocols that may require IRB review.

Sincerely,
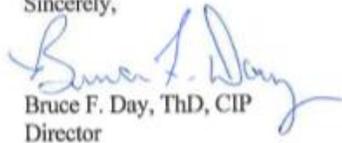
Bruce F. Day, ThD, CIP
Director

WE ARE... **MARSHALL.**
One John Marshall Drive • Huntington, West Virginia 25755 • Tel 304/696-4303
A State University of West Virginia • An Affirmative Action/Equal Opportunity Employer

29

APPENDIX B: SIMULATION SOFTWARE SOURCE CODE

```
Sbox = (
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B, 0xFE,
    0xD7, 0xAB, 0x76,
    0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4, 0xA2, 0xAF, 0x9C,
    0xA4, 0x72, 0xC0,
    0xB7, 0xFD, 0x93, 0x26, 0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71,
    0xD8, 0x31, 0x15,
    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2, 0xEB,
    0x27, 0xB2, 0x75,
    0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0, 0x52, 0x3B, 0xD6, 0xB3, 0x29,
    0xE3, 0x2F, 0x84,
    0x53, 0xD1, 0x00, 0xED, 0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A,
    0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F, 0x50,
    0x3C, 0x9F, 0xA8,
    0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5, 0xBC, 0xB6, 0xDA, 0x21, 0x10,
    0xFF, 0xF3, 0xD2,
    0xCD, 0x0C, 0x13, 0xEC, 0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64,
    0x5D, 0x19, 0x73,
    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14, 0xDE,
    0x5E, 0x0B, 0xDB,
    0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C, 0xC2, 0xD3, 0xAC, 0x62, 0x91,
    0x95, 0xE4, 0x79,
    0xE7, 0xC8, 0x37, 0x6D, 0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65,
    0x7A, 0xAE, 0x08,
    0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F, 0x4B,
    0xBD, 0x8B, 0x8A,
    0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E, 0x61, 0x35, 0x57, 0xB9, 0x86,
    0xC1, 0x1D, 0x9E,
    0xE1, 0xF8, 0x98, 0x11, 0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE,
    0x55, 0x28, 0xDF,
    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F, 0xB0,
    0x54, 0xBB, 0x16,
)

InvSbox = (
    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E, 0x81,
    0xF3, 0xD7, 0xFB,
    0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87, 0x34, 0x8E, 0x43, 0x44, 0xC4,
    0xDE, 0xE9, 0xCB,
    0x54, 0x7B, 0x94, 0x32, 0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42,
    0xFA, 0xC3, 0x4E,
    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49, 0x6D,
    0x8B, 0xD1, 0x25,
```

```
    0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xD4, 0xA4, 0x5C, 0xCC, 0x5D,
    0x65, 0xB6, 0x92,
    0x6C, 0x70, 0x48, 0x50, 0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7,
    0x8D, 0x9D, 0x84,
    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05, 0xB8,
    0xB3, 0x45, 0x06,
    0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02, 0xC1, 0xAF, 0xBD, 0x03, 0x01,
    0x13, 0x8A, 0x6B,
    0x3A, 0x91, 0x11, 0x41, 0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0,
    0xB4, 0xE6, 0x73,
    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8, 0x1C,
    0x75, 0xDF, 0x6E,
    0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89, 0x6F, 0xB7, 0x62, 0x0E, 0xAA,
    0x18, 0xBE, 0x1B,
    0xFC, 0x56, 0x3E, 0x4B, 0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78,
    0xCD, 0x5A, 0xF4,
    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59, 0x27,
    0x80, 0xEC, 0x5F,
    0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D, 0x2D, 0xE5, 0x7A, 0x9F, 0x93,
    0xC9, 0x9C, 0xEF,
    0xA0, 0xE0, 0x3B, 0x4D, 0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83,
    0x53, 0x99, 0x61,
    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63, 0x55,
    0x21, 0x0C, 0x7D,
)

xtime = lambda a: (((a << 1) ^ 0x1B) & 0xFF) if (a & 0x80) else (a << 1)

def text2matrix(text):
    matrix = []
    for i in range(16):
        byte = (text >> (8 * (15 - i))) & 0xFF
        if i % 4 == 0:
            matrix.append([byte])
        else:
            matrix[i // 4].append(byte)
    return matrix



def matrix2text(matrix):
    text = 0
    for i in range(4):  # 0~4
        for j in range(4):
            text |= (matrix[i][j] << (120 - 8 * (4 * i + j)))
    return text
```

```python
def __sub_bytes(s):
    for i in range(4):
        for j in range(4):
            s[i][j] = Sbox[s[i][j]]


def __inv_sub_bytes(s):
    for i in range(4):
        for j in range(4):
            s[i][j] = InvSbox[s[i][j]]


def __shift_rows(s):
    s[0][1], s[1][1], s[2][1], s[3][1] = s[1][1], s[2][1], s[3][1], s[0][1]
    s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2], s[1][2]
    s[0][3], s[1][3], s[2][3], s[3][3] = s[3][3], s[0][3], s[1][3], s[2][3]


def __inv_shift_rows(s):
    s[0][1], s[1][1], s[2][1], s[3][1] = s[3][1], s[0][1], s[1][1], s[2][1]
    s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2], s[1][2]
    s[0][3], s[1][3], s[2][3], s[3][3] = s[1][3], s[2][3], s[3][3], s[0][3]


def __mix_single_column(a):
    # please see Sec 4.1.2 in The Design of Rijndael
    t = a[0] ^ a[1] ^ a[2] ^ a[3]
    u = a[0]
    a[0] ^= t ^ xtime(a[0] ^ a[1])
    a[1] ^= t ^ xtime(a[1] ^ a[2])
    a[2] ^= t ^ xtime(a[2] ^ a[3])
    a[3] ^= t ^ xtime(a[3] ^ u)


def __mix_columns(s):
    for i in range(4):
        __mix_single_column(s[i])


def __inv_mix_columns(s):
    # see Sec 4.1.3 in The Design of Rijndael
    for i in range(4):
        u = xtime(xtime(s[i][0] ^ s[i][2]))
        v = xtime(xtime(s[i][1] ^ s[i][3]))
        s[i][0] ^= u
```

32

```python
            s[i][1] ^= v
            s[i][2] ^= u
            s[i][3] ^= v

    __mix_columns(s)

def encrypt(plaintext):
    plain_state = text2matrix(plaintext)

    __sub_bytes(plain_state)
    __shift_rows(plain_state)
    __mix_columns(plain_state)

    return matrix2text(plain_state)

def decrypt(ciphertext):
    cipher_state = text2matrix(ciphertext)

    __inv_mix_columns(cipher_state)
    __inv_shift_rows(cipher_state)
    __inv_sub_bytes(cipher_state)

    return matrix2text(cipher_state)

def chaos(x, y):
    line = encrypt(plaintext)

    sep = [int(digit) for digit in str(line)]
    k = len(sep)

    dot = []  # seed
    value = []
    x1 = []
    y1 = []
    x1.append(x)
    y1.append(y)
    dot.append(x * 2 + y * 2.5)
    for i in range(k):
        if i > 0:
            x1.append(y1[i - 1])
            y1.append(-0.2 * x1[i - 1] + 2.75 * y1[i - 1] - math.pow(y1[i - 1], 3))
            dot.append(x1[i] * 2 + y1[i] * 2.5)

    for j in range(k):
        random.seed(dot[j])
        num = random.randint(0, k)
```

```python
            value.append(sep[num])

        new_sep = ''.join('%s' % id for id in value)
        return new_sep


    def inv_chaos(text, x, y):
        sep = [int(digit) for digit in str(text)]
        k = len(sep)

        dot = []  # seed
        value = []
        x1 = []
        y1 = []
        x1.append(x)
        y1.append(y)
        dot.append(x * 2 + y * 2.5)

        for i in range(k):
            if i > 0:
                x1.append(y1[i - 1])
                y1.append(-0.2 * x1[i - 1] + 2.75 * y1[i - 1] - math.pow(y1[i - 1], 3))
                dot.append(x1[i] * 2 + y1[i] * 2.5)

        for j in range(k):
            random.seed(dot[j])
            num = random.randint(0, k)
            re = sep[j]
            value.insert(num, re)

        new_sep = ''.join('%s' % id for id in value)
return new_sep
```