Implementing and Testing a Novel Chaotic Cryptosystem

Samuel Jackson
Scott Kerlin
Jeremy Straub

Available at: https://works.bepress.com/jeremy_straub/303/
Implementing and Testing a Novel Chaotic Cryptosystem for Use in Small Satellites

Samuel Jackson
Oklahoma State University
Stillwater, OK
sam.jackson@okstate.edu

Scott Kerlin
University of North Dakota
Grand Forks, ND
skerlin@cs.und.edu

Jeremy Straub
University of North Dakota
Grand Forks, ND
jeremy.straub@my.und.edu

ABSTRACT

Cryptography in the domain of small satellites is a relatively new area of research. Compared to typical desktop computers, small satellites have limited bandwidth, processing power, and battery power. Many of the current encryption schemes were developed for desktop computers and servers, and as such may be unsuitable for small satellites. In addition, most cryptographic research in the domain of small satellites focuses on hardware solutions, which can be problematic given the limited space requirements of small satellites.

This paper investigates potential software solutions that could be used to encrypt and decrypt data on small satellites and other devices with similarly limited resources. Specifically, this paper presents an implementation of an encryption algorithm based on chaos theory and compares and contrasts it with existing cryptographic schemes used in desktop computing and lightweight applications.

Categories and Subject Descriptors
E.3 [Data]: Data Encryption

General Terms
Algorithms, Performance, Security

Keywords
Chaotic cryptosystem, small satellite, CubeSat, encryption, decryption, security, cryptography

1. INTRODUCTION

Small satellites are growing in popularity [9]. Programs such as the CubeSat Launch Initiative [7] in the United States and the European Space Agency’s Fly Your Satellite [4] program encourage students and researchers to build and launch small satellites. However, encryption is seldom used on small satellites. One reason for this is that FCC regulations on amateur radio bands ban any form of encryption [12]. However, as government, military, and commercial interest in small satellites grows, data confidentiality is becoming increasingly important. Many potential solutions are hardware-based and, while in many cases hardware-based cryptosystems give better performance, the mass and volume constraints of small satellites limit their usefulness and can preclude their use on these systems. There is a need for a reliable, software-based cryptosystem that is not computationally intensive, does not significantly increase bandwidth, and is energy efficient.

2. BACKGROUND

This section provides a brief overview of cryptographic research in the domain of small satellites, as well as in similar domains.

In 2012, Challa, Bhat, and Menair [2] proposed CubeSec and GndSec as a light-weight security solution for CubeSats, a specific type of small satellite. They proposed to use AES and DES encryption in Galois/Counter Mode on AES/DES supported hardware, specifically an ATX Mega 128 microcontroller. They achieved throughput of between 43KBps and 256Kbps depending on the electrical power available, however this work failed to investigate encryption techniques that rely solely on software.

In addition, there is a large body of work in the realm of wireless sensor networks, which have similar needs in regards to encrypting data with limited resources. There are many papers that investigate performance trade-offs between common algorithms such as AES and versions of RC4, RC5, or RC6 [6, 11]. Szczepaniak, et al. [10] tested whether Elliptic Curve Cryptography could be used to efficiently encrypt data between a network of sensor nodes and found it to be a viable solution. However, while public key cryptography lends itself well to a network made up of several sensors, a satellite needs only to communicate between itself and one (of a limited number of) ground stations at a time.

Lightweight cryptography is another burgeoning sub-field of cryptography, also focused in securing data confidentiality between devices with limited resources. In 2013, the National Security Agency published a paper recommending two new families of encryption algorithms: SIMON and SPECK [1]. The SIMON family of algorithms is optimized for hardware implementation, and the SPECK family is designed for software implementation, although both ciphers were designed for environments with limited hardware and software capabilities. Both of these algorithms perform better than many previous lightweight ciphers and are among the most optimized lightweight ciphers available.

3. A CHAOTIC BLOCK-BASED CIPHER

This paper proposes using a relatively new area of cryptography, chaos-based image encryption [3], to create a block-based cipher to provide the data confidentiality needed in small satellites. While this algorithm has not been as thoroughly tested as other, more common algorithms such as AES, it has the potential to be less computationally taxing on a system with limited resources. The algorithm, which is described below, is based on work by Huang, Ye, and Wong [5].
3.1 General Information
The chaotic cryptosystem presented in [5] was created specifically with the goal of encrypting images. An image is read into a two-dimensional array of size $n \times n$, where each index holds one pixel. For each diagonal and anti-diagonal row in the array, the position of each pixel in that diagonal or anti-diagonal is permuted. This disguises which pixels were originally next to each other (see Section 3.4). Finally, block-based diffusion is performed on the image, such that if a small change in the original image will result in a large change in the final image (see Section 3.5).

3.2 Initial Values and the Lorenz System
The set of values used throughout the algorithm is generated using the system of equations presented below. These are generally known as the Lorenz system.

\[
\begin{align*}
\dot{x} &= m(y - x) \\
\dot{y} &= rx - y - xz \\
\dot{z} &= xy - bz
\end{align*}
\]

(1)

For initial values $m = 10, r = 28,$ and $b = 8/3$, the system exhibits chaotic behavior. Hence, given initial values $x_0, y_0,$ and $z_0$, will soon diverge and start generating values that differ vastly from a system with similar initial values $x_1, y_1,$ and $z_1$. Given this, the Lorenz system can be used to generate secret values used in both the diagonal / anti-diagonal shuffling step and the block-based diffusion step. Given initial secret values $x_0, y_0,$ and $z_0$, the system of equation can be iterated. The system is first iterated $p$ times, so that it is given sufficient time to diverge from systems with similar starting values. For this work, $p = 30$. The system is then iterated an additional $n$ times, where $n$ is the size of one side of the array. Each $x$ value returned is placed in order into a one-dimensional array of size $n$. Similarly, the $y$ and $z$ values returned are placed into their own arrays. There are now three one-dimensional arrays of length $n$ that hold the values iterated by the Lorenz system. As the values of $x, y,$ and $z$ are generally increasing or decreasing along a given set of consecutive values, the terms are modified by the equation:

\[
m = \text{abs}(m \times 10^3) - \text{floor}[\text{abs}(m \times 10^3)]
\]

(2)

where $m \in \{x, y, z\}$, abs($m$) returns the absolute value of $m$, and floor($m$) returns the nearest integer less than or equal to $m$. We have now obtained three, one-dimensional arrays of length $n$, each with a seemingly random set of values. Note that this differs from the suggested equation in [3], which is:

\[
m = \text{abs}(m \times 10^4) - \text{floor}[\text{abs}(m \times 10^4)]
\]

(3)

3.3 Circulant Matrices
Previously, it was stated that the matrix must be permuted along the diagonal and anti-diagonal lines. However, if a square matrix (such as the one shown in Figure 1) is considered, it is apparent that the length of the diagonal lines differ. For example, the diagonal that runs from the top left vertex to the bottom right vertex has $n$ values, where $n$ is the length of a side of the array. The diagonal to the right has $n-1$ values. To solve this problem, the diagonals can be ‘wrapped’ to a corresponding diagonal so that the total diagonal has $n$ values, as shown in Figure 1.

\[
\begin{array}{ccc}
1 & 2 & 3 \\
3 & 1 & 2 \\
2 & 3 & 1
\end{array}
\]

Figure 1. An example of ‘wrapping’ diagonal lines so that each diagonal has $n$ values. Each number corresponds with a given wrapped diagonal row.

3.4 Performing Diagonal and Anti-Diagonal Shuffling
There is now one two-dimensional $n \times n$ array that holds the data, and three one-dimensional arrays which hold the adjusted iterated values. A copy is then created of the one-dimensional array which holds the $x$-values. This copy is sorted, giving a total of two arrays with the same values, but in a different order. It is now possible to map the pixels in a given diagonal line to the array of unsorted $x$-values. For example, the first pixel is mapped to $x_1$, the second pixel to $x_2$, and so on. The pixels are then permuted along the diagonal line so that it matches the sorted array of values. This operation is performed for each diagonal and anti-diagonal line, however anti-diagonal lines are permuted based on the difference between the array of $y$-values and sorted $y$-values.

\[
\begin{array}{cc}
\text{Unsorted Array: } \{x_1, x_2, x_3, x_4\} \\
\text{Sorted Array: } \{x_2, x_4, x_3, x_1\}
\end{array}
\]

Before Permutation

<table>
<thead>
<tr>
<th>1 (x_1)</th>
<th>2 (x_2)</th>
<th>3 (x_3)</th>
<th>4 (x_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

After Permutation

3.5 Block-Based Diffusion
Block-based diffusion must now be performed, with the goal that a small change in the original image will result in a large change of the cipher image. This process starts by separating the $n \times n$ array that holds the permuted pixels into two smaller arrays, of size $n/2 \times n$, designated $A_1$ and $A_2$. A simple way to do this is to take the top half of the array and place it in $A_1$, and take the bottom half of the array and place it in $A_2$. Next, choose $\frac{n^2}{2}$ values from the initial value arrays. It is important to note at this step that for any $n > 6$, we will need to select more values than there are unique values, and repetition will occur. Using these values, it is possible to create an array of size $N/2 \times N$, designated $D$. Each element in $A_2$ is summed together and used to calculate the value of a number $t$, given by the equation:

\[
t = (\sum A_2 \mod \left(\frac{n^2}{2}\right)) + (\sum A_2 \mod (n)) + (\sum A_2 \mod \left(\frac{n^2}{2}\right)) + 1
\]

(4)

where $\sum A$ is the sum of each element in $A$, and a mod($b$) applies the modulo($b$) operation to a. Each element in $D$ is multiplied by $t$. Finally, each element in $A_1$ is added to the corresponding element in $D$, and the operation modulo (256) is applied. Thus, the diffused version of $A_1$ is created, designated $\overline{A_1}$.

To diffuse the bottom half, $A_2$, each element in $A_2$ is added to the corresponding element in $\overline{A_1}$, and the modulo (256) operation is applied. $A_2$ is now diffused, and is designated $\overline{A_2}$. By applying the final two steps that calculate $\overline{A_1}$ and $\overline{A_2}$ multiple times, the cipher image is diffused. In this work, the cipher image is diffused five times. The resulting cipher image is the combination of $\overline{A_1}$ and $\overline{A_2}$. 
3.6 Expansion to Block Cipher
The original algorithm (developed by Huang, Ye, and Wong [5]) that this work is based on was created with the goal of encrypting images. However, it is possible to place virtually any data into a suitable matrix. Thus, this idea has been extended to create a new block cipher. In Section 3.5, it was required to choose $\frac{n^2}{2}$ values from the initial array values. In the implementation presented in this paper, a random number generator was used to select values, with the current time acting as the seed. Because of this, the starting time must be transmitted with the cipher text.

4. SMALL SPACECRAFT USE
The application of this algorithm to use onboard a small spacecraft is straightforward. However, testing is required to demonstrate its efficacy for this use. Testing must, thus, be conducted to demonstrate that the algorithm can perform suitably in an environment with limited resources.

It is unclear, from the literature, as to how well AES would perform in the context of a small spacecraft’s resource constraints. In addition, the NSA has suggested that the SPECK family of algorithms can be used as a comprehensive lightweight encryption scheme. Hence, testing will be conducted to characterize the performance of AES and SPECK and compare the algorithm proposed herein to it. Specifically, this testing will focus on the algorithm’s computational intensity and total throughput. All of these algorithms meet the requirement of not significantly increasing the total size of the cipher text (as compared to the plain text).

For the purposes of assessing the performance of these algorithms on a small spacecraft, a test plan (based on the computational hardware specifications of the OpenOrbiter spacecraft [8]) has been developed. OpenOrbiter utilizes two types of computational hardware: a Raspberry Pi computer serves as the primary flight computer (which is responsible for managing moment-to-moment operations) and GumStix WaterStorm computer-on-module (COM) units are used to create a payload processing center. The performance of the cryptographic system will be characterized on both types of hardware. This facilitates decision making regarding whether cryptographic processing can be performed on the flight computer or if the payload processing center must be utilized for this purpose, in the context of the OpenOrbiter mission’s testing and demonstration of this algorithm in low-Earth orbit. The testing of multiple algorithms on these two common processing systems may facilitate the decision-making of others.

5. CONCLUSION
There is a large gap in cryptographic algorithms where small satellites are concerned. Very little research has been done in this field, and what little has been done has been focused on hardware-based solutions. We have implemented a chaotic cryptosystem as a proof-of-concept, and also compared it to several existing well-known systems. In addition, we have demonstrated that it is possible to extend this algorithm to encrypt more than just pictures. As discussed in section 3.6, it is theoretically possible to extend this algorithm to encrypt any data, by placing it into a matrix format. In future work we plan to assess whether additional security can be provided by first hiding data in a picture using steganography, then applying this encryption scheme to the picture. A comparison of this hybrid technique to other existing data security systems is planned.

6. ACKNOWLEDGMENTS
This research was funded by the U.S. National Science Foundation (NSF Award # 1359224) with support from the U.S. Department of Defense.

7. REFERENCES