Testing a Novel Cryptosystem for Use in Securing Small Satellite Communications

Samuel Jackson[1], Scott Kerlin[2], Jeremy Straub[2]
[1]Department of Computer Science, Oklahoma State University
[2]Department of Computer Science, University of North Dakota

Introduction

Cryptography in the domain of Small Satellites is a topic of growing importance. While large satellites are likely to have the hardware requirements to run common cryptographic algorithms, small satellites are extremely limited in both hardware capabilities, which limits the speed and security of cryptosystems implemented in software, and available physical space, which limits the ability to include cryptosystems implemented in hardware. However, small satellites are growing in popularity, and as such securing communications becomes a necessity for some. The Department of Defense is exploring the possibility of using CubeSats, a type of small satellite, in their operations, as are branches of the U.S. military. This poster investigates a novel encryption scheme, with the goal of finding a software-based cryptosystem that has the capability to run on small satellites or larger, older satellites with outdated hardware.

Background

In 2012, CubeSec and GndSec were proposed as a light-weight security solution by Challa, Bhat, and Mcnair [1]. The proposal suggested using AES and DES encryption operating in Galois/Counter Mode on AES/DES supported hardware. Speeds of between 43KBps and 256Kbps were achieved using their method.

Lightweight cryptography is a developing area of cryptography, focused specifically on securing data in environments with limited resources. The NSA published two families of lightweight cryptographic algorithms in 2013, SIMON and SPECK [2]. Both algorithms are similar, however the SIMON family of algorithms is optimized for hardware while the SPECK family is optimized for software. These algorithms performed better than many other preexisting lightweight algorithms, and were suggested as potential standards of lightweight encryption algorithms.

Finally, it is important to note potential issues in securing communications to and from small satellites, specifically CubeSats. Many CubeSats use amateur radio frequencies, and FCC regulations state that encryption of data is prohibited on those bands [3]. Other approaches to communication are possible, however.

Algorithm

The proposed algorithm is a proof-of-concept based on the algorithm presented by Huang, Ye, and Wong [4]. It was created with the intent to encrypt and decrypt image files. The work of Huang, Ye, and Wong has been extended for use as a general block cipher.

In essence, the algorithm reads data into an *n* x *n* matrix. It then uses a system of equations known as the Lorenz System to generate values based off an initial key set {x, y, z}. The matrix of data is then permuted along diagonal and anti-diagonal lines. Finally, as adjacent pixels are normally similar, each pixel value is diffused using a block based diffusion method. Figure 1 shows the general outline of the algorithm

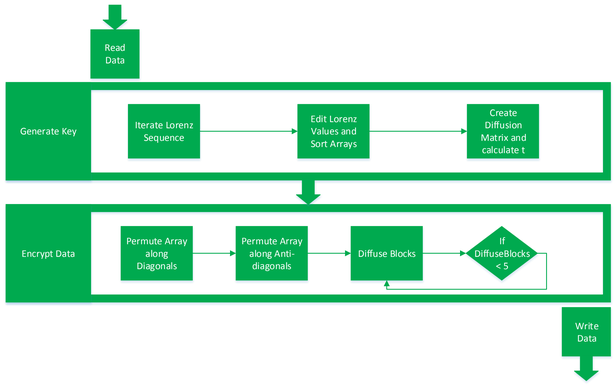

Figure 1: A chart showing the general design of the novel algorithm

Testing

Testing was performed on a Raspberry Pi computer. The proposed algorithm was compared against implementations of AES (using the Crypto++ Library) and SPECK. In addition, multiple block sizes of the proposed algorithm were tested, namely *n* = 4, 8, 16, 32, 64, 128, and 256, where *n* is one side of an *n* x *n* matrix. Block sizes for the proposed algorithm are 16, 64, 256, 1024, 4096, 16384, and 65536 bytes respectively. Six test files were used, one text file, three picture files of different sizes, and two movie files of different sizes. Each file was encrypted and decrypted by each algorithm a total of 150 times, in groups of 50 at a time.

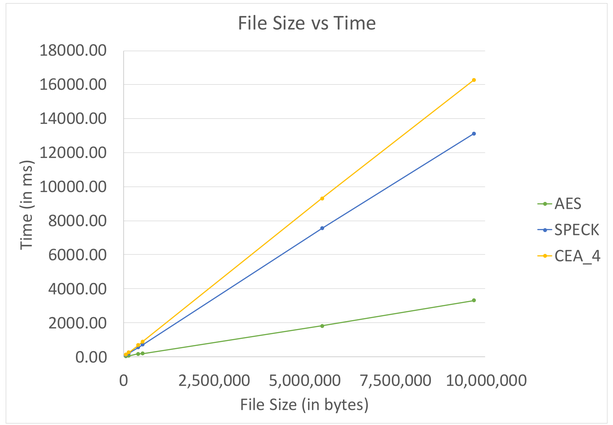

Figure 2: A chart showing the comparison between encryption times of AES, SPECK, and the 16 block size of the propose algorithm

Algorithm:		Text File	Small Picture	Medium Picture	Large Picture	Small Movie	Large Movie
AES	Mean	122.21	155.67	208.36	218.67	1896.95	3369.14
	Median	26.44	59.07	167.11	192.11	1820.96	3307.72
SPECK	Mean	133.48	229.60	563.94	768.55	7565.59	13180.98
	Median	92.53	199.89	545.28	717.17	7558.81	13131.20
16 byte	Mean	247.08	321.14	721.23	938.66	9376.04	16380.62
	Median	116.39	249.53	677.43	890.02	9310.50	16276.60

Table 1: Encryption time comparison between the novel algorithm and AES and SPECK. Times shown are in ms.

Algorithm:		Text File	Small Picture	Medium Picture	Large Picture	Small Movie	Large Movie
16 byte	Mean	247.08	321.14	721.23	938.66	9376.04	16380.62
	Median	116.39	249.53	677.43	890.02	9310.50	16276.60
64 byte	Mean	235.02	280.08	630.58	820.12	8162.89	14142.78
	Median	181.70	214.81	584.57	768.05	8079.60	14058.80
256 byte	Mean	294.86	376.05	615.42	770.55	7944.10	13910.27
	Median	94.40	202.35	549.64	721.93	7707.64	13622.30
1024 byte	Mean	216.49	261.31	588.56	771.28	7735.40	13503.79
	Median	96.97	206.13	556.55	730.85	7686.17	13365.95
4096 byte	Mean	611.32	396.03	641.75	811.43	7726.50	13618.76
	Median	586.87	205.07	543.04	714.36	7490.01	13251.30
16384 byte	Mean	524.18	539.65	590.94	780.36	8044.89	14309.31
	Median	551.14	343.87	555.32	730.54	7804.75	14235.70
65536 byte	Mean	370.34	475.52	961.50	1217.91	10073.63	14139.14
	Median	294.81	306.69	679.57	865.36	8117.33	14030.40

Table 2: Results of testing of the novel algorithm. Times shown are in ms.

Acknowledgements

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

Results

Results are presented in Figure 2 and Tables 1 and 2, showing encryption times of AES, SPECK, and the proposed algorithm. Table 1 shows fairly clearly that AES is the fastest algorithm when running on a Raspberry, being almost five times faster than SPECK. While SPECK is faster than the proposed algorithm in all tests, the difference between their times are reasonably close.

In regards to the different block sizes of the proposed algorithm, block sizes of 256, 1024, and 4096 appear to perform the fastest, although block sizes of 64 and 16384 were either comparable or only marginally slower The proposed algorithm with a block size of 65536 blocks ran quite a bit slower for small file sizes, but the difference in processing time decreased as file size increased. Conversely, a block size of 16 performed marginally slower in smaller files, and the difference in times increased as file size increased.

Conclusion / Future Work

Results show that, when running on a Raspberry Pi, AES was consistently the fastest algorithm to encrypt and decrypt files. However, it is important to note that the Raspberry Pi includes hardware optimization for AES, and the Crypto++ library is able to take advantage of this increase. Testing should be done without hardware optimization to compare the proposed algorithm against AES.

In addition, while SPECK was faster than the novel algorithm, it may be possible to optimize the novel algorithm further, surpassing the speeds of SPECK.

References

1. Challa, O., Bhat, G., and Mcnair, J. CubeSec and GndSec: A Light-weight Security Solution for CubeSat Communications in Small Satellite Conference (Logan, UT, 2012), DigitalCommons@USU, 8 pages
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. *Cryptology ePrint Archive, 2013 (404)*, 45 pages
3. Title 47 CFR 97.113 (a)(4). Prohibited Transmissions. U.S. Government Publishing Office. July 7, 2015.
4. Huang, X., Ye, G., and Wong, K. Chaotic Image Encryption Algorithm Based on Circulant Operation. Abstract and Applied Analysis, 2013 (2013), 8 pages