A comparison of conventional cryptosystems

Matthew Lee
University of South Carolina Upstate
Computer Science Senior Seminar
matthew-c-lee@outlook.com

ABSTRACT

Society runs on secrets. When sending emails, using an ID to enter a building, or even using a debit card, encryption is used to keep your information safe. Without encryption, data can be easily read and used to maliciously gain access to systems and wreak havoc. The development of strong encryption methods ensures that our society can continue to function.

There are many different methods of encryption, some of which are more easily cracked than others. There is a trade-off between security, speed, processor usage, power consumption, and memory utilization.

This research aims to explore the differences between two popular symmetric encryption techniques: Substitution Permutation Networks and Stream Ciphers.

Keywords

Symmetric Encryption, Block Ciphers, Substitution Permutation Network, Synchronous Stream Cipher, Pseudorandom Keystream

1. INTRODUCTION

Encryption is the act of converting data into inscrutable information by transforming the raw data using a mathematical function called a cipher.

The main way of encrypting data before the advent of DES, was by using stream ciphers. Some popular forms were using rotor machines and military hardware such as linear-feedback shift registers [6].

Stream ciphers generally use a finite state machine which is initialized with the key to create a pseudo-random string, that is then XORed with the plaintext to create the ciphertext.

The arrival of the Data Encryption Standard (DES), created by IBM, marked cryptography's shifting focus from stream ciphers to block ciphers [6]. Block ciphers encrypt plaintext blocks by mixing them with a fixed key in such a way that it is (ideally) incapable of being brought back to plaintext without the correct key.

For almost three decades, the Federal Information Processing Standards' (FIPS) standard encryption cipher was the DES. It worked using a Feistel Cipher, which is a form of block cipher. With Feistel Ciphers, encryption and decryption are very similar functions, with a "round function" being ran iteratively a certain number of times to transform the data.

After it became clear that exhaustive key search could crack DES [7], there were several remedies explored by the cryptographic community. One solution was to instead use Triple-DES, which resulted in a much stronger algorithm, but was also three times slower. While this solution saw much use, the NIST began a contest to find a new standard block cipher in 1997. The winner would be proclaimed the Advanced Encryption Standard (AES).

Three years later and after much evaluation, Rijndael was named to be AES due to its "security, performance, efficiency, ease of implementation, and flexibility"

The switch from DES to AES also marked the move away from Feistel Networks. The AES is a substitution-permutation network (SPN) cipher [6]. It is made up of layers of S-boxes (substitution boxes), and allows for key sizes of 128, 192, and 256, with 10, 12, and 14 rounds, respectively.

As the standard encryption technologies throughout the history of symmetric data encryption have been stream ciphers, block ciphers using Feistel Networks, then SPN ciphers, those are the ones I aim to compare in terms of their speed, strength, and memory usage.

2. LITERATURE REVIEW

[4] looked at the performance of a few popular algorithms – namely, DES, Triple DES, AES, and Blowfish. They used Java (JDK 1.4) to implement the algorithms because the compiler produces byte code rather than machine code, ensuring that the programs can run on any platform with the Java interpreter. Java's BigInteger class was used to manipulate integer values of keys and data blocks.

They found that getting consistent, repeatable execution times was rather difficult. In the end, the researchers decided to exclude the initialization and key setup times from the comparison. Since

decryption time is almost equal to encryption time for all algorithms, only encryption times were measured.

Their results found Blowfish as the fastest, with DES, AES, and Triple-DES behind it, respectively. Their results were consistent with the idea that the larger the block size, the faster the algorithm.

[4] used CFB mode to test the Stream Cipher performance. They found a reversal of the previous rule – the larger the block size for Stream Ciphers, the slower the algorithm. They reasoned that this is due to the algorithm having more work for the same amount of input data in a single algorithm execution cycle. A smaller block size would allow the same size of input data to be encrypted more efficiently.

One factor was consistent between block ciphers and stream ciphers – a larger key slows down encryption since a larger number of bits are involved in each execution cycle.

Their research concluded that Blowfish was the fastest algorithm, but they did not consider memory usage or security in their analysis.

Measuring the security of ciphers has proven to be a difficult task for the cryptographic community. There are two different paradigms to consider a cipher's security. One is the knowledge that it is secure against all known cryptanalytic attacks. This, of course, is subject to change, but is nonetheless vital to the confidence in the algorithm. Another way to inspire confidence in a cipher is through mathematically provable security. This method is also one that can be used with statistical methods, which will be useful when comparing algorithms to one another.

One popular method of cracking encrypted data is through differential cryptanalysis, looking for a lack of uniformity in the output, which gives clues about key bit information [5]. Patterns between plaintext, key, and output are analyzed to crack the codes.

[3] describes a statistical test to discover the randomness or lack thereof in a block cipher, by "systematically constructing one bit input differences". A low score on the randomness test is an indication of a lower quality cipher.

Diffusion is a property that describes the plaintext bit's chance to affect the output's bits. High diffusion indicates a higher number of output bits affected by an input bit.

Under the proposal in [3], a matrix is calculated using the calculated diffusion instances – that is, a snapshot of the diffusion capacity of a cipher.

For a potentially strong cipher, one would expect that the number of ones would be equal to the number of zeroes. This is due to the confusion of a cipher, where the goal is that the chance of an output bit's inversion is 0.5 given the inversion of the input bit. The frequency test is ideal for this. [1] The other criteria are that they should be randomly distributed in the matrix, and that the matrices Ψ_i and Ψ_j are not similar for $i \neq j$ [3].

Essentially, they compare actual and estimated values to determine whether a cipher acts as a random source, and the zeroes are placed randomly in the matrices, or whether there is some consistency in the placement of them [3]. They used DES to test their method of determining randomness because it has been shown to be a non-pseudorandom function. [2]

Product	Expected	Actual	Difference (%)	diff_rand_test( )
$Q_1 \times Q_2$	0.241739	0.216797	2.5	FAIL
$Q_1 \times Q_3$	0.204115	0.179688	2.4	FAIL
$Q_1 \times Q_4$	0.126188	0.077148	4.9	FAIL

Figure 1. DES differences from randomness test [3]

The previous research into both the cryptographic strength and encryption/decryption speed has been extremely useful, but an analysis of these factors together on multiple ciphers could improve our knowledge of these cipher's usage.

3. METHODOLOGY

This section concerns the methodology behind the experiment, looking into the rationale for SPN ciphers, stream ciphers, and Claude Shannon's metrics for a secure cipher.

3.1 Substitution-Permutation Network Methodology

A substitution-permutation network takes a block of plaintext along with a block of the key as its inputs.

In essence, it applies alternating rounds of mathematical operations, namely substitution and permutation, to produce the ciphertext from the plaintext. Decrypting with this method only requires reversing the process.

Figure 2. Substitution-permutation network with 3 rounds [?]

The rounds of substitution rely on a substitution box, or S-box, to function. The S-box substitutes each block of bits it is given with the output of the S-box. It ensures that it cannot be inverted without the key by substituting one-to-one. The substitution should also have an avalanche effect of change, such that each round's output contributes to the next round's input, resulting in a wildly different total output.

The P-box is a permutation of the bits. It works by taking the output of the S-box, permuting the bits, then feeding them into the next round's S-box. This should result in the output bits of the S-box being distributed to many S-box inputs.

Each round, the key schedule, which is generated from the key, is used to XOR the data, preventing a reversal of the process without the key.

3.2 Stream Cipher Methodology

A stream cipher can be contrasted with a block cipher. While the latter performs computations on "blocks" of a certain number of bits at a time, a stream cipher's computations go "bit by bit".

Figure 2. Keystream generator operations [10]

With a stream cipher, plaintext bits are combined with a pseudorandom digit stream, called the keystream, to generate the output. They are combined by encrypting each plaintext bit one-at-a-time with the corresponding keystream bit, adding the resulting bit to the ciphertext stream. The typical mathematical operation used for combining is the processor efficient XOR function.

Stream ciphers are typically faster and have lower hardware complexity.

There are two different types of stream ciphers – synchronous and self-synchronizing stream ciphers. Synchronous stream ciphers have the pseudorandom digits generated independently, then are combined with the plaintext for encryption or decryption.

3.3 Diffusion and confusion

Confusion and diffusion are identified by Claude Shannon to be properties of a secure cipher [9]. Both are important factors in preventing a cipher from being broken by methods of cryptanalysis.

We will be looking at and measuring the Diffusion of both stream ciphers and SPN ciphers. Having a good amount of diffusion means that upon changing a single bit of the input text, about half of the resulting ciphertext bits are expected to change. This is to hide the relationship between the input and output. Half of the bits changing every time ensures that patterns in the plaintext are not apparent in the ciphertext.

Therefore, we can measure diffusion by analyzing the difference in outputs of two inputs with single-bit differences. The percentage of bits changed (Hamming Distance) will be multiplied by 100 to obtain the diffusion score.

4. IMPLEMENTATION

For this implementation I used Python 3.10, along with the timeit and scipy libraries for the defaulttimer() and and hamming() functions respectively. Defaulttimer() allowed for taking a snapshot of the exact time the function was run, which enabled the calculation of how long between certain snippets of code – namely, encryption and decryption functions. Scipy's hamming function, given two arrays as its input, outputs the hamming distance between them. To use this only requires both the plaintext and ciphertext's bitstrings to be converted to arrays.

4.1 SPN Permutation Implementation

The permutation function takes in the S_box and block of bits as inputs. The result is initialized as 0 before the loop. The loop iterates, starting at the value of the block size (16 in my case) and decreases itself to zero. Each iteration, the result is increased by the value of the input bytes modded by 2, then shifted left by the block size decreased by p_box[i]. The bytes are then shifted to the right by one place. All of this is done to shift the bits around in a pseudorandom fashion.

In my implementation, each round the permuted bits are XORed with the corresponding iteration of the key schedule. The permuted bits are then used for substitution.

The key schedule is generated from the secret key by repeatedly creating a subkey that is equal to the secret key's current value modded by 12, then removing 4 bits from the end of the secret key. This generates a unique key schedule of length equal to the set number of subkeys.

4.2 SPN Substitution Implementation

The substitution function works by initializing the result as 0, then determines the spot in the s_box array it will look by modding the bytes in the block by the block size. That s_box value will then be reduced by the iteration of the loop we are in, then added to what will be the output of the function. Then the block's bytes are shifted to the right 4 places so that we can get meaningful values from the next piece of the block. This process is repeated four times to substitute the entire 16-bit block of data.

4.3 Stream Cipher Implementation

The stream cipher encrypts data using a key. The key is then used

as the seed for the Python random library, which then creates a pseudorandom keystream. The keystream is set to be the same size as the plaintext. The plaintext is then XORed with the keystream, resulting in the encrypted data. Decryption is a simple reversal of this process.

5. DATASET AND EXPERIMENT SETUP

This section is for giving insight into the design and setup of how the data was collected. It aims to show explanations of the evaluation metrics chosen, as well as the system used to create the dataset.

The metrics that we will be looking at are Encryption Time, Decryption Time, and Diffusion.

We can repeat experiments many times since encryption is a relatively quick process. For both measuring encryption and diffusion, we will repeat the experiment 10,000 times and take the average to ensure that it is not skewed by outliers.

5.1 SPN Experiment Setup

The steps to run the experiment are relatively simple. These steps are already written and working in each cipher Python file, but I will outline them here. For the Substitution-Permutation Network, the first step is to determine a secret key in binary. This key can be any permutation of zeroes and ones, but it must be of a 32-bit length, given that the block length of the algorithm is 16-bits.

We must also initialize the number of subkeys, which will determine the length of the key schedule.

Next, determine a plaintext length. This will determine the speed of the algorithm – a chunk of text ten times the size of another will also be ten times as slow when encrypting and decrypting. Therefore, we must use the same chunk of plaintext on both ciphers.

The encryption and decryption times are determined with the measure_all_timings() function, which takes as inputs the secret key, number of subkeys, the plaintext, as well as n – the number of times we want the experiment to run. It returns an array as such: [encryption_average, decryption_average].

Diffusion is calculated similarly. Function get_average_diffusion() takes the same inputs, but returns only the diffusion average as the output.

We then output the results.

5.2 Stream Cipher Experiment Setup

To perform these experiments on the Stream Cipher, we first create a key. Then create the plaintext, which must be equal to the one we used for the SPN cipher.

Diffusion is determined with get_average_diffusion(), which takes the key, plaintext, and number of experiment runs.

The get_average_timings() function also takes the key, plaintext, and number of experiment runs, but returns an array of [encryption_average, decryption_average].

Again, we output these results. The resulting diffusion and timing data will be used in our final analysis of the data.

6. RESULTS

The results of our comparison between our Stream Cipher and Substitution-Permutation Network are listed in this section.

Plaintext used:

"Lorem Ipsum is simply dummy text of the printing and typesetting industry."

Stream Cipher			
Iterations	Average Encryption Time	Average Decryption Time	Diffusion
10,000	0.0134	0.0133	52.7027

Substitution-Permutation Network Cipher			
Iterations	Average Encryption Time	Average Decryption Time	Diffusion
10,000	0.4852	0.8526	49.1554

7. ANALYSIS

Comparing these results, the Stream Cipher takes much less time to encrypt and decrypt than the SPN Cipher. In fact, it is over 36 times as fast.

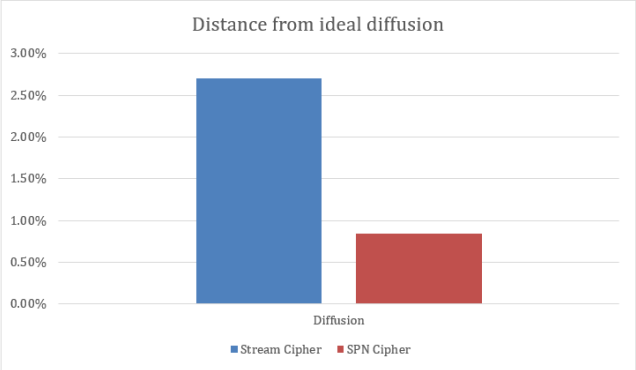

Figure 3. Distance from ideal diffusion

The Stream Cipher achieves 52.7% diffusion, which is higher than the ideal of 50%, with a difference of 2.7%. This means that it does not have perfect diffusion.

The SPN Cipher has a diffusion of 49.16%, which is within a point of ideal diffusion. This is a good indication that the SPN is an improvement upon the Stream Cipher in terms of encryption strength. The difference is 0.84%.

Another interesting result is the large gulf between the encryption time and decryption time with the SPN Cipher. While average encryption and decryption time in the Stream Cipher are about the same, the SPN Cipher's decryption time takes about 1.76 times longer than encryption time. Through using the timeit library, we were able to discover that the added decryption time is due to the key schedule permutations.

Our findings conclude that the Stream Cipher we used was much faster in terms of encryption and decryption than the Substitution-Permutation Cipher we built. At the same time, the SPN Cipher was found to be a better encryption algorithm in the view of Claude Shannon's diffusion metric.

We would conclude that for mission-critical encryption use cases where time is not a huge factor, a Substitution-Permutation Network architecture such as used in this experiment would be preferred. On the other hand, if speed is more important and security is less important, a Stream Cipher like the one used here may be adequate.

8. CONCLUSIONS

In this research, I explored the difference between two of the most popular symmetric encryption techniques: substitution-permutation networks and stream ciphers.

These results allowed me to discover the upsides and downsides of each. In short, SPN ciphers are more secure according to the diffusion metric but take much longer to encrypt and decrypt.

One flaw in this research is that I did not test the algorithms against the various types of cryptanalysis, of which there are dozens of different methods.

What makes this research unique is that I directly analyzed the differences between two different types of encryption that I implemented myself, rather than the differences between two commonly-used ciphers such as Blowfish, AES, DES, etc.

REFERENCES

[1] Knuth, Donald Ervin. The art of computer programming. Vol. 3. Pearson Education, 1997.

[2] Luby, Michael, and Charles Rackoff. "How to construct pseudorandom permutations from pseudorandom functions." SIAM Journal on Computing 17.2 (1988): 373-386.

[3] Katos, Vasilios. "A randomness test for block ciphers." Applied mathematics and computation 162.1 (2005): 29-35.

[4] Nadeem, Aamer, and M. Younus Javed. "A performance comparison of data encryption algorithms." 2005 international Conference on information and communication technologies. IEEE, 2005.

[5] Biham, Eli, and Adi Shamir. "Differential cryptanalysis of DES-like cryptosystems." Journal of CRYPTOLOGY 4.1 (1991): 3-72.

[6] Biryukov, Alex. "Block ciphers and stream ciphers: The state of the art." Cryptology EPrint Archive (2004).

[7] Diffie, Whitfield, and Martin E. Hellman. "Exhaustive cryptanalysis of the NBS data encryption standard." Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman. 2022. 391-414.

[8] (Wikimedia Foundation. (n.d.). Substitution-permutation network. Wikipedia. Retrieved October 22, 2022

[9] Anderson, D. R. (2008). Information theory and entropy. *Model based inference in the life sciences: A primer on evidence*, 51-82.

[10] (Wikimedia Foundation. (n.d.). Stream cipher. Wikipedia. Retrieved October 22, 2022