Block Ciphers: Basics of Design and Cryptanalysis

CSCI-200 Independent Study Project

Khoa Nguyen

May 1, 2016

Abstract

This paper introduces basic concepts and definitions in cryptography and cryptanalysis, with a strong concentration on block ciphers. We define and explain several components of modern block ciphers, briefly cover some popular cryptanalytical techniques used on them, and provide a proof-of-concept implementation of a substitution-permutation network as defined later in Section 2.1.2. Block ciphers are very important in modern data encryption, but they are not commonly taught at the undergraduate level. One reason is that the study of cryptography and cryptanalysis in general requires a strong background in theoretical mathematics. Thus, this paper also serves to show what one needs in order to do further research in this field.

Contents

List of Figures

Listings

1 Overview

Cryptography is the study of keeping information secure by encryption. Cryptanalysis is the study of crypto systems, ciphers, and ciphertexts in order to find weaknesses in them and retrieve the original plain messages. Two main branches of key-based encryption algorithms are public-key and private-key (also called symmetric or conventional algorithms). Private-key encryption schemes are further divided into stream ciphers and block ciphers. This paper concentrates on defining, implementing, and cryptanalyzing block ciphers with specific examples of existing encryption algorithms.

1.1 Cryptography

Humans have wanted to protect their private conversations since the earliest communication. From Caesar's cipher to the RSA, cryptography has gradually become an extensive field with increasingly essential applications and profound effects on everyone's lives. In particular, the development of cryptography explodes with the rise of computer and computer networks. Today, cryptography has two main branches: private-key and public-key. Private-key, also called symmetric encryption schemes include stream ciphers and block ciphers, which are mostly applied in data encryption.

We denote encryption and decryption of a symmetric algorithm with $(E_K(M) = C, D_K(C) = M)$, where E_K is the encryption scheme, D_K is the decryption scheme, M is the original message, and C is the resulting ciphertext. In most cases of symmetric algorithms, key K is the same for both encryption and decryption.

The ultimate goal of cryptography is *unconditional security*, or *perfect secrecy*, as defined below:

Definition 1.1. Perfect secrecy. A cipher provides **perfect secrecy**, or unconditional security, if the ciphertext and plaintext blocks are statistically independent [6].

Theoretical results on perfect secrecy provide motivation for research and design of better crypto systems. However, in practice, it is rather impractical and unnecessary due to the current limited computing power. For example, in block ciphers' case, identical ciphertext blocks imply identical plaintext blocks; thus perfect secrecy cannot be obtained by a block cipher encrypting more than one block. If a secure block cipher is used, the costs of changing the key and redoing all the computations required before encryption for every block far outweigh the little, if any, improvement in security.

1.1.1 Block Ciphers

Block ciphers operate on groups of bits (usually 64 bits) called blocks. Notable block ciphers include AES (Advanced Encryption Standard), the Twofish family, and DES (Data Encryption Standard). Often, the goal of cryptanalytic attacks against block ciphers and other private-key encryption schemes is to discover the key.

Despite the existence of public-key encryption such as the RSA, block ciphers, notably the AES, are still essential for the protection of data. To encrypt a large amount of data almost instantly as required by many software applications and network transactions, block ciphers are preferred over RSA because of their speed and efficiency in terms of hardware. The entire security of block ciphers and other private-key crypto systems lies in the chosen key. Private-key and public-key encryption schemes are thus often combined to form hybrid crypto systems, in which the private-key scheme encrypts the data, then the public-key scheme encrypts the respective key to maintain security during the communication [7]. Additionally, block ciphers can also form fundamental building blocks in pseudo-random number generators, stream ciphers, hash functions, and some digital signature schemes [6].

1.2 Cryptanalysis

As mentioned above, cryptanalysis is the study of crypto systems, ciphers, and ciphertexts in order to find weaknesses in them and retrieve the original plain messages. In designing any crypto system, cryptanalysis must be used to prove the relative se-

curity of the new system against all conceivable attacks thus far. A good crypto system designer, therefore, is often also a good cryptanalyst who has seen enough weaknesses of other systems to avoid them, and is objective enough to attempt breaking their own design.

A common assumption to make in cryptography is that everyone, including cryptanalysts and attackers, have complete knowledge of how the crypto system is designed, i.e the entire secrecy and confidentiality lies in the key. The cryptanalyst may employ one or more of the following types of attacks in order to find the key [7]:

- **Ciphertext-only:** Given ciphertexts $C_1, C_2, ..., C_n$ from the same encryption scheme, the cryptanalyst finds the corresponding plaintexts $P_1, P_2, ..., P_n$; and/or key K; and/or an algorithm to infer P_{n+1} from a new ciphertext C_{n+1}.
- **Known-plaintext:** Given plaintexts $P_1, P_2, ..., P_n$ and their corresponding ciphertexts $C_1, C_2, ..., C_n$ from the same encryption scheme, the cryptanalyst finds key K; and/or an algorithm to infer P_{n+1} from a new ciphertext C_{n+1}.
- **Chosen-plaintext:** This is similar to the known-plaintext attack, but in this case the cryptanalyst can choose certain plaintexts to be encrypted that might reveal more about the key K.
- **Chosen-ciphertext:** The cryptanalyst chooses certain ciphertexts to be decrypted to ultimately find the key K.
- **Rubber-hose:** The cryptanalyst threatens, blackmails, tortures, or bribes someone to get the key from them. This is often the best way to break an algorithm.

In contrast with the popular notion that the "acceptable" way to break an encryption algorithm is to use ciphertext-only attack, the other types of attacks are employed quite frequently. The security of an algorithm is considered trustworthy if all the details of its design are accessible, yet no attack is possible.

1.3 Motivation and Outline of Approach

The rest of this paper is devoted to the definition, related concepts, and certain examples of block ciphers. Despite the important role that block ciphers play in data encryption, sometimes they do not receive much emphasis in established general computer science and mathematics programs, especially at the undergraduate level. This paper serves to introduce block ciphers, cover the basics that one needs before conducting deeper research in this branch of cryptography, and implement a simple version of substitution-permutation networks, a popular structure used in many current block ciphers.

2 Block Ciphers: Introduction and Cryptanalysis

This section provides a more detailed definition based on essential components of a block cipher, several concepts surrounding its implementation, popular cryptanalytical techniques, and a few examples of notable block cipher algorithms.

2.1 Definition and General Concepts

We define block ciphers as below:

Definition 2.1. Block cipher. A block cipher is a function that maps n-bit plaintext blocks to n-bit ciphertext blocks, i.e for every key $K \in \{0,1\}^k, E_K : \{0,1\}^n \to \{0,1\}^n$ is a permutation, and both E_K and $D_K = E_K^{-1}$ can be computed quickly given K, where k is the key's length and n is the block's length.

Recall from the previous section that perfect secrecy in practice is very costly and impractical. Instead, we consider *computational security* [6]:

Definition 2.2. Computational security. A block cipher is considered **computationally secure** if:

1. The block size is sufficiently large to preclude exhaustive data analysis. Given a data complexity of 2^n, or the expected number of distinct input data units, in this case blocks of bits, the encryption function of a fixed k-bit key can

be completely characterized. Thus we need to choose a block length n large enough to make processing 2^n inputs computationally prohibitive.

2. The effective key size is sufficiently large to preclude exhaustive key search. Given a processing complexity of 2^k, or the expected number of operations required to process input data, an attack can be carried out by exhaustive search among all possible key combinations.

3. No known attack has both data and processing complexity significantly less than 2^n and 2^k respectively.

2.1.1 Finite Fields and Galois Fields

This section provides some preliminary mathematical background needed to understand the current design of block ciphers. Many modern block ciphers, including DES, AES, and Twofish, are designed based on knowledge of finite fields and polynomials over a field. We define these concepts as below:

Definition 2.3. Fields and Finite Fields. F is a **field** if F has well-defined operations denoted $(+)$ and $(\cdot)$ such that:
(a) $\forall a, b \in F : a + b \in F$
(b) $\forall a, b, c \in F : (a + b) + c = a + (b + c)$
(c) $\forall a, b \in F : a + b = b + a$
(d) $\exists \mathbf{0} \in F, \forall a \in F : a + \mathbf{0} = a$
(e) $\forall a \in F, \exists b \in F : a + b = \mathbf{0}$
(f) $\forall a, b \in F : a \cdot b \in F$
(g) $\forall a, b, c \in F : (a \cdot b) \cdot c = a \cdot (b \cdot c)$
(h) $\exists \mathbf{1} \in F, \forall a \in F : a \cdot \mathbf{1} = a$
(i) $\forall a, b, c \in F : (a + b) \cdot c = (a \cdot c) + (b \cdot c)$ (j) $\forall a, b \in F : a \cdot b = b \cdot a$
(k) $\forall a \neq \mathbf{0} \in F, \exists b \in F : a \cdot b = \mathbf{1}$
F is a **finite field** if F has a finite number of elements [11].

Definition 2.4. Polynomials over a field F. A **polynomial with coefficients**

in F is an expression of the form

$$b(x) = b_n x^n + b_{n-1} x^{n+1} + ... + b_1 x + b_0,$$

where x is called the *indeterminate* of the polynomial, and the $b_i \in F$ are called *coefficients*. The **degree** of a polynomial is the exponent on the highest power of x having a nonzero coefficient.
The set of polynomials over a field F is denoted $F[x]$. The set of polynomials over a field F whose degrees are less than l is denoted $F[x]|_l$ [11][2].

We define addition of polynomials as consisting of summing the coefficients with equal powers of x where the addition of the coefficients is defined the same way as it is in the field F. Multiplication in $F[x]|_l$, however, is a little more tricky because the product needs to be contained in $F[x]|_l$ as well, i.e the multiplication $(\cdot)$ operation is closed. We define the multiplication of two polynomials $a(x)$ and $b(x)$ as the algebraic product $a(x) \cdot b(x)$ modulo a *reduction polynomial* $m(x)$ of degree l where $m(x)$ is *irreducible*, which means there exist no two polynomials $g(x), h(x) \in F[x]|_l$ with positive degrees such that $m(x) = g(x) \cdot h(x)$. Irreducible polynomials in a field are like prime numbers in the set of integers. An irreducible polynomial cannot be written as a product of two or more polynomials with positive degrees in the field other than **1** and itself, the same way a prime number cannot be factored into a product of two or more positive integers other than 1 and itself.

The simplest field, and one that is frequently used in various ciphers, is $\mathbb{Z}_2 = GF(2) = \{0, 1\}$ (GF stands for Galois Field after the famous mathematician's name), containing only 2 elements. It is obvious that operation $(+)$ in $GF(2)$ is XOR $(\oplus)$ [5] when considering 0 and 1 as bit values. Also, addition and subtraction are the same operation in this field, since $1 + 1 = 0$ $(mod\ 2)$ and $1 = -1$ by definition, thus $1-1 = 1+(-1) = 1+1 = 0$. Storing a polynomial can be done by placing its coefficients together in a string. If $l = 8$, then a polynomial in $GF(2)|_8$ can be stored as an 8-bit value, or a byte, and abbreviated by the corresponding hexadecimal notation. For example, the polynomial in $GF(2)|_8$ $x^5 + x^2 + x + 1$ corresponds to 00100111, or 27 in hexadecimal notation. Now we define addition and multiplication in $GF(2)|_8$

as outlined above. Choosing an irreducible polynomial $m(x) \in GF(2)|_8$ for the definition of multiplication renders $GF(2)|_8$ a field, often denoted $GF(2^8)$. We also say that $GF(2^8)/m(x)$ is a representation of the field $GF(2^8)$.

For example, consider $GF(2^8)/(x^8 + x^4 + x^3 + x + 1)$. The product of the polynomials denoted by the hexadecimal values 57 (0101 0111) and 83 (1000 0011) is the polynomial denoted by C1 (1100 0001), because

$$(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) \tag{1}$$
$$= (x^{13} + x^{11} + x^9 + x^8 + x^7) \oplus (x^7 + x^5 + x^3 + x^2 + x) \oplus (x^6 + x^4 + x^2 + x + 1) \tag{2}$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \tag{3}$$
$$\equiv x^7 + x^6 + 1 \ (mod\ x^8 + x^4 + x^3 + x + 1). \tag{4}$$

On line 2, we expand the multiplication by multiplying $(x^6+x^4+x^2+x+1)$ with each term in $(x^7 + x + 1)$, and denote the addition operation between the three resulting polynomials as XOR ($\oplus$). Performing the XORs results in line 3. The final result on line 4 is obtained by doing the following long division in $GF(2^8)$ to get the remainder $x^7 + x^6 + 1$:

$$\begin{array}{r|cccccccccc}
 & & & & & & +x^5 & & +x^3 & & \\
\hline
x^8 + x^4 + x^3 + x + 1) & +x^{13} & +x^{11} & +x^9 & +x^8 & & +x^6 & +x^5 & +x^4 & +x^3 & +1 \\
 & +x^{13} & & +x^9 & +x^8 & & +x^6 & +x^5 & & & \\
\cline{2-8}
 & & +x^{11} & & & & & & +x^4 & +x^3 & +1 \\
 & & +x^{11} & & & +x^7 & +x^6 & & +x^4 & +x^3 & \\
\cline{2-10}
 & & & & & +x^7 & +x^6 & & & & +1
\end{array}$$

2.1.2 Substitution-Permutation Networks

Many recent block ciphers, including Data Encryption Standard (DES) and Advanced Encryption Standard (AES), employ some variant of a simple type of iterated cipher called *substitution-permutation network*.

Definition 2.5. Substitution-Permutation Networks. A substitution-permutation

network (SPN) is a product cipher composed of a number of stages called *rounds*, each involving substitutions and permutations [6] [10]. Suppose a block of plaintext has $n = lm$ bits where $l, m \in N$. An SPN has two components π_S and π_P such that:

- $\pi_S : \{0,1\}^l \rightarrow \{0,1\}^l$ is called an S-box (S stands for "substitution") and replaces l bits with a different set of l bits.
- $\pi_P : \{0,1\}^{lm} \rightarrow \{0,1\}^{lm}$ is a permutation and permutes lm bits.

Given an n-bit block of plaintext X where $n = lm$, we can break X up into m substrings $x_1, x_2, ..., x_m$, each of which has l bits. In each round of the SPN, we perform one (or more) π_S on each of the above x_i, then perform one π_P on the whole block of input bits X. For better security, variants of SPNs often include many rounds in the SPN, more than one S-box, larger S-boxes, and different or key-dependent S-boxes in each round.

Even though the concept of an SPN is relatively easy to understand, usage of a strong permutation function π_P and especially a strong substitution function π_S (i.e S-box) can greatly improve the security of a block cipher algorithm. For example, Rijndael (the chosen candidate for AES) and Twofish (a finalist in the selection of AES) both apply field theory in designing their key schedules, permutations, and S-boxes to decrease the linearity of plaintext-ciphertext pairs, thus increasing the complexity costs to attack these ciphers successfully. A proof-of-concept implementation of an SPN is outlined in Section 3.

2.1.3 Modes of Operation

Block ciphers encrypt fixed-size blocks of bits in the plaintext. Often, the block size is $n = 64$ bits. For longer messages, the cipher needs certain specifications, called *modes of operation*, on how to encrypt such input plaintexts. Security concerns have led cryptographers to design several different approaches. At present, there are four popular modes of operation for block ciphers [2]:

- Electronic Code Book (ECB) mode: This is the natural, intuitive solution, which is to break the longer plaintexts into n-bit blocks and apply the cipher on

them in the same way. However, this has some obvious security disadvantages, such as the same blocks of plaintexts yielding the same blocks of ciphertexts.

- Cipher Block Chaining (CBC) mode: This is proposed as an alternative to ECB mode. Before applying the cipher on each input plaintext block p_i, it is XORed (exclusive OR logic operation) with the ciphertext block corresponding with the previous plaintext block p_{i-1}. When decrypting the ciphertext, each ciphertext block c_i goes through the inverse cipher, then an XOR with the previous ciphertext block c_{i-1}. If the size of the last plaintext block is less than n, both ECB and CBC modes have to pad it till it has size n, or add some kind of complexity to the cipher only for that last block.

- Cipher Feed Back (CFB) mode: Both CFB and OFB modes are called key-stream generation modes. Here, the block cipher is used to generate a key stream, which is bitwise XORed or subtract-XORed with the plaintext stream for encryption and decryption respectively. In CFB mode, the key stream is a key-dependent function of the last b bits of the ciphertext.

- Output Feed Back (OFB) mode: The block cipher in this case is used to generate the key stream based on a finite number of states having its block length and updated by a key-dependent updating function. Counter mode is a special case of OFB mode, where the key-stream is generated by "applying ECB-mode encryption to a predictable sequence, e.g an incrementing counter."

2.2 Popular Cryptanalytical Techniques against Block Ciphers

This section describes the general idea of linear and differential cryptanalysis. Unfortunately, explaining the probability theory and algorithms behind these techniques would require another 10-20 pages to be written which far exceeds the limits placed on this paper. Thus, additional details about these techniques and examples of their usage can be found in [10][7][6].

2.2.1 Linear Cryptanalysis

Linear cryptanalysis is popular in block cipher cryptanalysis because theoretically, this approach applies to any iterated cipher. Suppose that there is a probabilistic linear relationship between a subset of plaintext bits and a subset of intermediate bits before the substitutions performed in the last round. If an attacker carries out a known-plaintext attack, i.e they have a large amount of plaintext-ciphertext pairs encrypted by the same unknown key K, they can begin decrypting the ciphertexts by trying all the possible keys that might have been used in the last round. For each of those keys, we test if the above probabilistic relationship holds. If it does, we increase that key's frequency counter. Hopefully by the end of the process, the key with the highest frequency count contains the correct bit values of the unknown key $K[10]$.

2.2.2 Differential Cryptanalysis

Differential cryptanalysis is very similar to linear cryptanalysis, except that differential cryptanalysis compares the XOR of two plaintext blocks of bits to the XOR of the two corresponding ciphertext blocks of bits. Thus, this is a chosen-plaintext attack. In other words, we compute the fixed XOR value x' of the two plaintext blocks x_1 and x_2, whose corresponding ciphertext blocks are y_1 and y_2 respectively. Then we decrypt y_1 and y_2 using all possible candidate keys for the last round of the cipher. For each of those keys, we test if the XOR values of certain intermediate bits has the most likely value for the fixed x'. If it does, we increase that key's frequency counter. Hopefully by the end of the process, the key with the highest frequency count contains the correct bit values of the unknown key $K[10]$.

2.3 Examples of Block Ciphers

2.3.1 Advanced Encryption Standard (AES)

In 1997, the National Institute of Standards and Technology (NIST) announced an open selection process to choose a successor cipher, AES, to replace the old DES. In the end, Rijndael, developed by two Belgian cryptologists, Vincent Rijmen and Joan

Daemen, was chosen to become AES with a few minor modifications to Rijndael's variable block length and key length. The AES fixes the block length to 128 bits, and supports key lengths of 128, 192, and 256 bits only. Rijndael is a key-iterated block cipher, consisting of the repeated application of a round transformation on the state. The number of rounds depends on the block length and key length [2]. The general structure of Rijndael is like below:

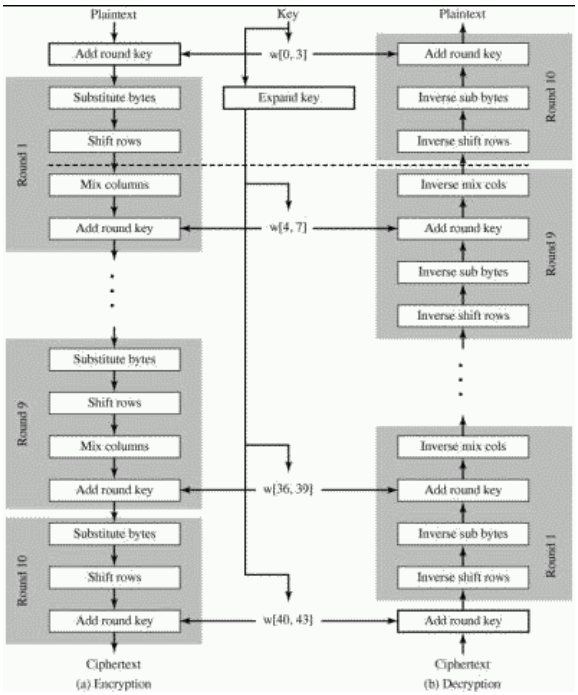

Figure 1: Structure of AES [4].

2.3.2 Twofish

Twofish is a block cipher algorithm in the top five finalists being considered to become AES. Unlike AES, Twofish uses a structure called a Feistel network, a general method of transforming any function into a permutation [9]. Another special feature of Twofish is that it uses a 4-by-4 MDS (maximum distance separable) matrix as the final piece in its key schedule to produce the subkeys and the pre-computed S-boxes for each of its 16 rounds. A MDS code over a field is a linear mapping from a field

elements to b field elements, producing a composite vector of $a + b$ elements where the minimum number of nonzero elements in any nonzero vector is at least $b + 1$. Because MDS code is linear, it can be represented by an MDS matrix consisting of $a \times b$ elements. An $a \times b$ matrix is an MDS matrix if and only if all possible square submatrices, obtained by discarding rows or columns, are non-singular, i.e invertible [9].

In the computation of the MDS matrix, $GF(2^8)$ is represented as $GF(2^8)/(x^8 + x^6 + x^5 + x^3 + 1)$. Then

$$a(x) = a_7x^7 + a_6x^6 + ... + a_1x + a_0 = \sum_{i=0}^{7} a_ix^i \in GF(2^8)$$

corresponds to the byte value

$$a_7a_6a_5a_4a_3a_2a_1a_0 = \sum_{i=0}^{7} a_i2^i.$$

In the original Twofish paper, the MDS matrix is given by

$$MDS = \begin{bmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{bmatrix}.$$

These values are not random, but carefully computed over the field $GF(2^8)/(x^8 + x^6 + x^5 + x^3 + 1)$, and chosen to facilitate faster implementation [3].

3 Case Study: Implement a Substitution-Permutation Network (SPN)

This section outlines the proof-of-concept implementation of a simple substitution-permutation network (SPN) as previously defined in Section 2.1.2. We define the

block length to be 8 bytes, or 64 bits, and the key length to be 16 bytes, or 128 bits, as standard in most modern block ciphers. Each byte in the key is randomly generated, ranging from 0 to 255, corresponding to 00 to FF in hexadecimal notion respectively. This SPN is implemented as a class in C++, including:

- One 8×8 S-box π_S.
- One permutation function π_P.
- A key schedule that generates an 8-byte subkey for each round of the SPN.
- Encrypt and decrypt functions.
- Other private helper functions.

The pseudo-code of its main encryption algorithm is:

SPN(*plaintext*)

```
1  w_0 := plaintext
2  for i := 1 to n − 1
3        x_i := w_{i−1} ⊕ k_i
4        s_i := π_S(x_i)
5        p_i := π_P(s_i)
6        w_i := p_i
7  x_n := w_{n−1} ⊕ k_n //Last round
8  s_n := π_S(x_n)
9  ciphertext := s_n ⊕ k_{n+1}
10 return ciphertext
```

Listing 1: Encryption algorithm of the SPN.

In this algorithm, the x's save the result after each XOR, k_i is the subkey of round i, the s's save the result after each substitution π_S, and the p's save the result after each permutation π_P. In the last round, π_P is not applied. Instead, the result after the last π_S is "whitened" by being XORed with an extra subkey, k_{n+1}. Decryption can be done by reversing the steps above with appropriate inverse functions of π_S and π_P.

Below is a visualization of the structure of this SPN, where n is the number of

rounds specified in the initialization of an instance of this SPN (the default number used in test files is $n = 8$):

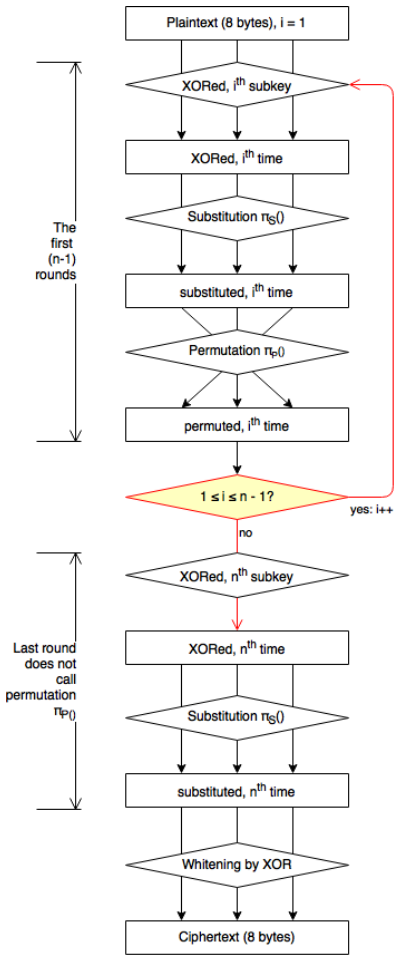

Figure 2: A visualization of the SPN's algorithm.

It is clear that the concept of an SPN is very simple, and most modern block ciphers follow a similar structure. What makes an SPN into a strong encryption algorithm is mainly how the permutation and substitution functions are implemented such that it immensely reduces the probability of discovering the key with less resources than a brute force attack or exhaustive search, as in AES and Twofish.

3.1 SPN 1.0

3.1.1 Key schedule

From the 16-byte key, each 8-byte subkey k_i of round i is generated simply by copying 8 bytes of the original key starting from byte $3i + 1$ and wrapped around to the beginning of the key again in case of overflow. This is not a secure key schedule and only good for illustrative purposes.

3.1.2 S-box π_S

The S-box receives an 8-byte-long input and returns an 8-byte-long output. Each byte is substituted by its bit-flipped version, i.e 0000 0001 (0x01) becomes 1111 1110 (FE). One security drawback of this S-box is that it is susceptible to a linear attack due to the strong linear relationship between the input bytes and the output bytes.

3.1.3 Permutation π_P

The permutation function receives an 8-byte-long input and returns an 8-byte-long permuted version of that input. The 8×8 permutation matrix and its inverse are pre-computed when we call the SPN object's constructor. The permutation matrix is guaranteed to be invertible because its 8 columns are the standard basis of $\mathbb{R}^8$, i.e $\vec{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ ... \\ 0 \end{bmatrix}, \vec{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ ... \\ 0 \end{bmatrix}, ..., \vec{e}_8 = \begin{bmatrix} 0 \\ 0 \\ ... \\ 0 \\ 1 \end{bmatrix}$, in a randomly permuted order. When π_P is called, it interprets the 8-byte input as a vector of length 8, and multiplies this vector with the permutation matrix. The output vector thus has the same entries as the input vector but in a different permutation. The randomization of the order of the vectors $\vec{e}_i$'s in the permutation matrix is implemented such that the positions of the entries in the input vector never stay the same.

3.1.4 Analysis

The above definitions of π_S and π_P render SPN 1.0 susceptible to linear attacks because there is a strong linear relationship between the value of the input bytes and the output bytes, as shown in the comparison between the original image and the encrypted one:

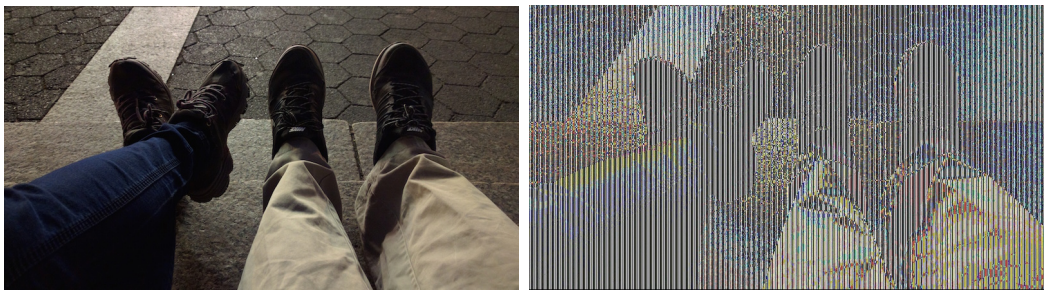

Figure 3: Original image versus encrypted image.

When random plaintexts are generated and graphed against their corresponding ciphertexts, the correlation between them looks even clearer, showing that this SPN is not very secure against a linear attack:

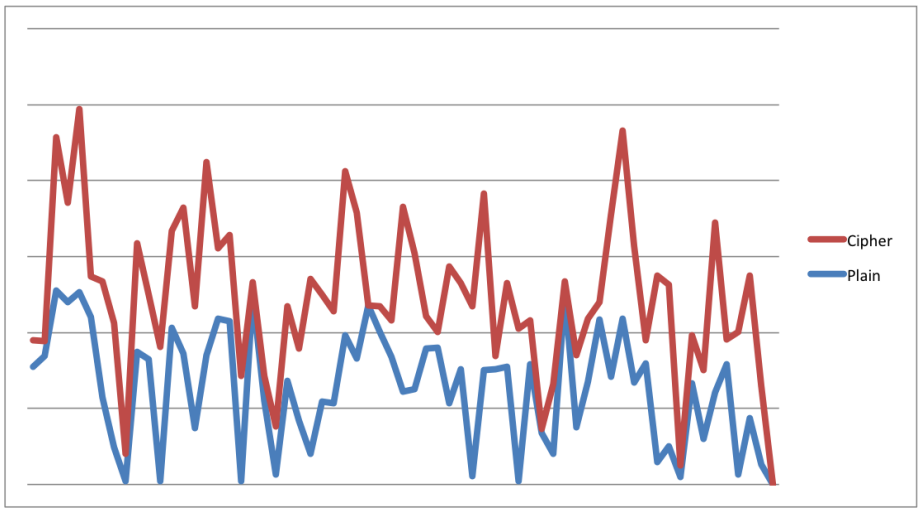

Figure 4: Plaintext versus ciphertext.

3.2 Future Improvements

Recall the definition of perfect secrecy in Section 1.1 that the ciphertext and plaintext blocks need to be statistically independent. Even though trying to be perfect is impractical, later versions of this SPN should definitely be improved, especially its key schedule and substitution function, in order to reduce the linear correlation between the values of the ciphertexts and plaintexts. Possible improvements include:

- Apply field theory and number theory in creating the substitution and permutation functions.
- Implement the algorithm in another programming language or paradigm.
- Perform cryptanalytical attacks against the existing algorithm to find and fix its weaknesses.

4 Conclusion

This paper introduces basic concepts in cryptography and cryptanalysis, and devotes a majority of its contents to defining and explaining several aspects of block ciphers. Future work includes fully implementing an SPN in a different programming paradigm, and to continue to research possible attacks against existing block ciphers currently in use such as AES and Twofish. It is obvious that a cryptographer or cryptanalyst needs a very strong background knowledge of theoretical mathematics in order to fully understand, analyze, and design modern ciphers. Cryptography remains one of the most important fields of study in computer science and mathematics. As computing power increases and memory storage gets larger, concerns over the protection of information grow stronger. Thus, to study how encryption schemes work and how to design them should be a priority of every programmer and computer scientist.

References

[1] Boneh, Dan, and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2015. https://crypto.stanford.edu/~dabo/cryptobook/draft_0_2.pdf.

[2] Daemen, Joan, and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[3] FiloSottile (GitHub user). "Bindings for the Twofish Implementation by Niels Ferguson." Web, Accessed 10 April 2016. https://github.com/keybase/python-twofish.

[4] Firat, Murat. "Securing Data in .NET." Web, Accessed 30 April 2016. http://www.codeproject.com/Articles/21076/Securing-Data-in-NET.

[5] Garrett, Paul. "Class notes of MATH-5251: Error-correcting codes, finite fields, algebraic curves." Web, Accessed 08 April 2016. http://www.math.umn.edu/~garrett/coding/Overheads/08_crcs.pdf.

[6] Menezes, Alfred, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press LLC, 1997.

[7] Schneier, Bruce. *Applied Cryptography*. John Wiley & Sons, Inc., 1996.

[8] ———. "Twofish Source Code." https://www.schneier.com/cryptography/twofish/download.html.

[9] Schneier, Bruce, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. "Twofish: A 128-Bit Block Cipher." https://www.schneier.com/cryptography/paperfiles/paper-twofish-paper.pdf.

[10] Stinson, Douglas. *Cryptography: Theory and Practice*. Chapman & Hall/CRC, 2002.

[11] Wagstaff, Samuel Jr. *Cryptanalysis of Number Theoretic Ciphers*. CRC Press LLC, 2003.