The Revaris Block Cipher

A New Block Cipher Utilizing Feistel Network, hash-based key rotation, and 4 S-boxes

Dariel Valdano - 13512079
Program Sarjana Teknik Informatika
Institut Teknlogi Bandung
Bandung, Indonesia
13512079@std.stei.itb.ac.id, mail@dalva.net

***Abstract*—This Paper details a new block cipher algorithm called The Revaris Cipher, which utilizes a feistel network, utilizing 4 different S-boxes selected depending on key, and key-derived column and row transformation and multiple rounds to increase encryption strength. Key scheduling is performed by secure hashing part of the key depending on round number. The Revaris Cipher uses a block size of 512 bit and a key size of 512 bit.**

Keywords—Revaris Cipher, Feistel Network, S-Box, Encryption Algorithm, Block Cipher

I. Introduction

Block cipher encryption is one of the most common cipher types that is useful for encrypting and decrypting data. One early standard of a block cipher algorithm, the Data Encryption Standard (DES), relies on Feistel network and a Feistel function that contains an expansion stage, key mixing stage, substitution and permutation. This provides a so-called Confusion and Diffusion, a concept identified by Claude Shannon [1]. The DES algorithm has now been superseded by the Advanced Encryption Standard (AES) due to the small keyspace of the DES algorithm (56 bit) leading to the feasibility of a brute force attack. In 1998 the Electronic Frontier Foundation (EFF) created a custom DES-cracker that could crack a DES-encrypted message in just under 3 days [2].

In 2001, the Rinjdael algorithm, invented by Joan Daemen and Vincent Rijmen, was selected as the AES contest winner. The AES cipher, as it is now called, has a key size of 128, 192 or 256 bits [3]. Brute force attack is now infeasible for the AES, but other kinds of attack vectors was found. One example is the Related Key attack, discovered by Alex Biryukov and Dmitry Khovratovich [4]. This attack exploits the rather simple key scheduling mechanism of AES, and reduces the time requirement to crack relative to brute force. However, is still far beyond today's computational capabilities to crack in rational amount of time.

The weaknesses in AES' key scheduling mechanism can possibly be solved by using a more sophisticated way to generate the round key from the master key. Furthermore, AES' single S-box can be expanded to 4, selected by a pseudo-random number generator seeded by the round key. And finally, AES' row and column mixing can also be improved by using similar pseudo-random generated shuffler seeded by the round key.

II. Base of Theory

A. Block Cipher

Block Cipher is one method of encrypting and decrypting data that works on a block-by-block basis, the other being Stream Cipher. Each block consists of an array of bytes with predefined length, which will be operated on individually in the encrypt/decrypt function [5]. The output or input of this encrypt/decrypt function can then be used as an additional "key" for the next block using Cipher Block Chaining method of operation. As of the completion of this paper, AES is the most commonly used symmetric key block cipher algorithm.

B. Cipher Block Chaining

Cipher Block Chaining is a method of chaining each block of data by XOR-ing the previous ciphertext block to the current plaintext block before being encrypted [6]. As such, each ciphertext block relies on all other data blocks processed up to that point. This method of operation will also require an Initialization Vector, as an XOR-ing source for the first block of data.

Figure 1 shows a Cipher Block Chaining method in encryption (top) and decryption (bottom) modes.

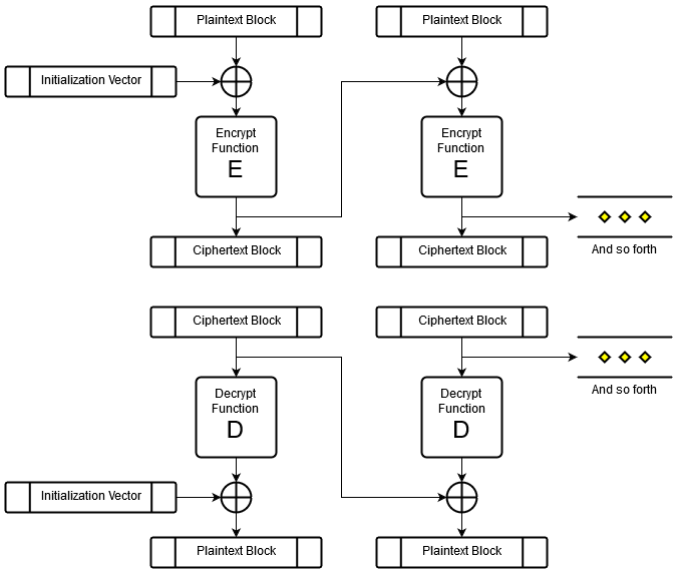

Figure 1 – Cipher Block Chaining

C. Feistel Network

Feistel Network is a structure used in block cipher, described first by Horst Feistel while working on the Lucifer Cipher. It is almost always performed in rounds, with each round containing the following operations [5]:

1. The round input is split into two halves L_{i-1} and R_{i-1},
2. The round function f_i is applied to the right half yielding $f_i(R_{i-1})$,
3. The Exclusive OR of this value and the left half is computed yielding $L_{i-1} \oplus f_i(R_{i-1})$, and
4. the left and right side are swapped, thus yielding the round output $L_i \,||\, R_i := R_{i-1} \,||\, L_{i-1} \oplus f_i(R_{i-1})$

Figure 2 shows how one round of Feistel network is performed. This round is then repeated multiple times, enhancing the cryptographic strength. The process of encryption and decryption is similar in structure, the only difference being the key scheduling.

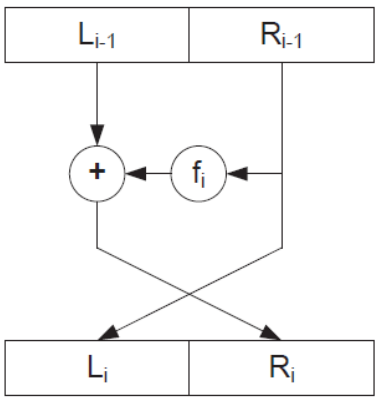

Figure 2 – 1 round of Feistel network [5]

D. Transformations and S-Box

Modern ciphers such as the Advanced Encryption Standard (AES) not only relies on confusion, but also on diffusion by shifting the rows of block state matrix and mixing the columns [3]. It also uses an S-Box substitution, which substitutes specific bytes in a data block according to a predefined Substitution Box, hence the name S-Box. Both if this methods, when performed in multiple rounds and combined with a proper key scheduling mechanism, strengthen the cryptographic quality of a cipher algorithm significantly.

E. Key Scheduling

In order to make sure that unique keys are used for each transformation and substitution round, as well as the Feistel round, a key scheduling method is used. Key scheduling method varies from as simple as taking a subset of key from a long string of main key, bitwise rotation of a key, to using one-way functions to expand a subset of the main key for the round key.

F. Secure Hash Algorithm (SHA)

SHA is a cryptographic hash function designed by the United States National Security Agency (NSA). SHA is considered to be a one-way function, due to the fact that reversing the function is considered to be practically impossible. Bruce Schneier considers it to be one of the workhorses of modern cryptography [7]. The function takes an input of arbitrary-length stream of byte, and outputs a fixed-length digest of the input. SHA-0 and SHA-1 is considered to be insecure, due to theoretical attacks and possible collisions. The newer version such as SHA-256, SHA-512 and SHA-3 is relatively more secure.

III. The Revaris Cipher Algorithm

The Revaris Cipher will utilize the Cipher Block Chaining method to chain each individual encrypted blocks. When processing each block, the Revaris' outer structure will utilize Feistel Network consisting of 32 rounds of feistel function. The feistel function will then contain yet another 32 internal rounds of column shuffling, row shuffling, and substitution using one of 4 S-boxes which will be selected based on the round key. Key Scheduling is performed in both the feistel round and internal round.

Note that in order to decrypt data, simple reverse the procedures listed in each level.

A. Revaris CBC Level

This is the outermost layer of the feistel cipher, where data management, key generation, Insertion Vector (IV) generation and block chaining is performed.

First, incoming bytes of data is padded if necessary to the nearest 512-bit ceiling. The padding contains a single 1-byte followed by 0-byte until the rest of the padding. For example, a 512-bit data is encrypted as is, but a 513-bit data will be padded to 1024-bit, with 1 in byte 514 and the remaining, byte 515 to 1024 filled with zeroes. This allows the identification of the padding itself when decryption is performed.

After the data is padded, the Master Key is generated from a key string inputted by user by using the SHA-512 algorithm. This will produce a consistent 512-bit Master Key from arbitrary length key that user inputs. Then, the IV is generated by salting the Master Key with 3 bytes of 0xFF, then hashed again with the SHA-512.

Finally, the data will be operated on 512-bit blocks, performing the CBC operation as illustrated in Figure 1.

B. Feistel Network Level

This level is symbolized as function E (and D when decrypting) in figure 1. This level consists of the Feistel Network, Feistel Function, and its supporting structures such as key scheduler, and XOR functions, and half-block rotator.

Once a block has been passed to the E function, it is first divided into 2 sub-block of the same length. Then, for each round, the right sub-block is passed inside the Feistel Function, and the output XOR-ed with the left sub-block. Afterwards the two sub-blocks are then rotated and the next round is ready to

be performed. The illustration of this process is available as Figure 2.

Each Feistel Round uses their own unique key by salting the master key with the feistel round number, then hashing it with SHA-256. The resultant key is then passed to the Feistel Function in each round.

After the feistel rounds are complete, the resultant block is then returned to the Revaris CBC Level to be used to XOR the next block before processing.

C. *Feistel Function and Internal Rounds Level*

The last and lowest level of the algorithm is the Feistel Function, which also incorporates an internal 32-rounds of row and column shuffling, and also byte substitutions. Figure 3 shows the structure of this level.

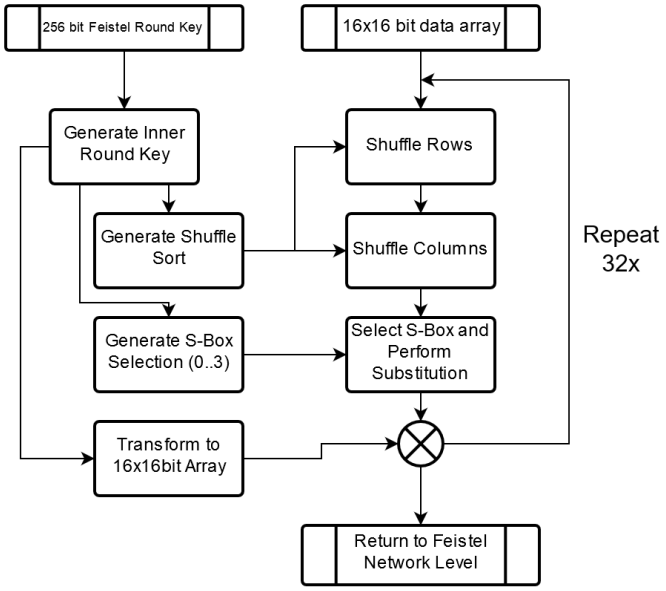

Figure 3 – Inside the Feistel Function

First, a 2 dimensional array of 16x16 bits were created and filled with the input data of the Feistel Function which is the 256-bit half-block. Then the key scheduling is initiated by rotating the key depending on rhe current round number. After the round key is set, the shuffle sort is generated using pseudo-random generated seeded with the round key, and the row and columns are shifted according the the generated shuffle sort.

Afterwards, the 16x16 bit array is transformed into 2x16 byte array, and each byte undergoes substitution using 4 possible S-boxes selected by the round key. Finally, the data array is then XOR-ed with the round key and then transformed back into a linear array of bytes to be returned to the Feistel Network Level for the next Feistel Round.

IV. Cipher Strangth Analysis

This cipher is analyzed in 4 different tests. The first is Random Data Test, to ensure that the generated ciphertext is not at all similar to the plaintext. The second is Diffusion test, to see the level of diffusion gained by encrypting a stream of 0xFF. The third being Key Sensitivity test, to see if the decryption key differs by one bit with the encryption key would make the decrypted plaintext legible.

A. *Random Data Test*

The following table shows the result of encrypting a random data. It is shown that the ciphertext is not at all similar compared with the plaintext. When compared, only 1009 out of 8192 bits has the same value.

TABLE I. Random Data Test

Plain Text (binary data):
5b2d2abf2157aa8236f6ead8d0fc97ef2f7e1a3fa5598c450beb0f208eee66a2 84864f1e8127917852f999417c3831073362b265a948fbbd8dfa9b302592cf55 2311f8d79fd293ed10e5120a905ec402dbb7d1e94437854aa675b4003d771461 cccb6c94547f51ba506a1fc8fd4dd635c780d483f06fcabb749e3aa7495a873c 099d6026013bb8c34cb30316c5d5bc95b634a0d9dee14e69d30db19a6e47f4df a87dfe197305e4135c24f3dc9879064371aeda39c089e0a1bee76bc6b068e3e8 8b5d631cf5962ee62276c91d42b91b08ddac15ce7b32285fc2ec4bafa3468a2b 8fb5ff7ae2cd172958a40c0e18539c677270c1f756f1ab6d2c044064ad88f23e
Key (string): "test"
Cipher Text (binary data):
f4e5e9e4badacd24d0f77abb911512cda80c500461a66cf5f49d155d28bc92ff 756aa8a575fce67b0c21ae87846deaebf4b6d56967eca54fc4cb4e20ab727484 e8298b65d84a7a11c1925d8c3b396341d5ff3a42271938aaadcf48f4c7858ee0 485d3b58f1b8cabe26bb9b2275a1b3639d20bc0035ec28b075ff4cc062fe3c6b f2af029ab50c66b838cace24d787c7085173901075e3e835fa156b6b1c5182dd aae7d0e127efb10ecaa82c2e2fee5294f25df190d1517252b6f8764f1cd74c3a 8f6bc512ba5bf4c9538b39b87b8fd282af8c8ee52b7ed1cd02625381b3ee362a c7b7a0ee7ed3916856a13e697797735c250a5384e7c84e20dcc80cff0c132bb6

B. *Diffusion Test*

The diffusion quality of the algorithm is tested by encrypting a 4KB binary data that are filled with ones. By performing frequency analysis of the resulting ciphertext by counting the occurrence of unique 8-bit sequences, the diffusion quality can be determined. The more uniform the frequency graph, the better the diffusion quality of the algorithm is.

The following figure shows the frequency chart, sorted by value:

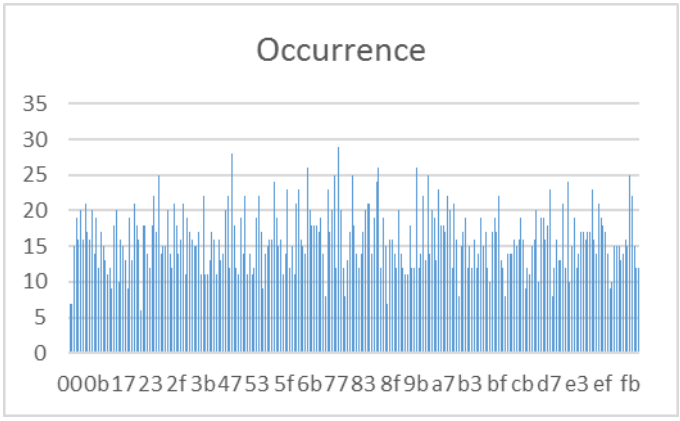

Figure 4 – Frequency Chart by value

The following figure shows the same chart sorted by number of occurrence:

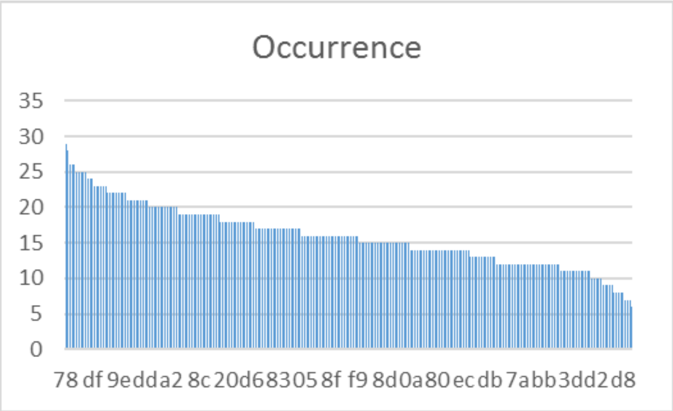

Figure 5 – Frequency Chart by number of occurrence

C. *Key Sensitivity Test*

The final test is the Key sensitivity test, which will compare 2 ciphertexts of the same data encrypted with key A and decrypted with key B, with key B differs with key A by a single bit.

The following table shows the result of this test, with 2048 bits out of 8192 bits between encrypted binary data and decrypted binary data being the same.

TABLE II. KEY SENSITIVITY TEST

Plain Text (binary data):
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
Encryption Key (string): “test”
Cipher Text (binary data):
ea0d296fcacd57ebfeb19e024622fb8195a24cb9e76459a7704af87244db6663 c57f4d63060a02d2a70aa758ddb6f6346e8b8e48750deca78f3efcf38dc7990c 8ce821b5c376ce2dac7515afa22909d103f120a74bfba054efd32dd1ff9fd7b2 db897ef4c85a6dc9dd1418ca0fca080167e78ff481958c3b75c2d86365420d7f 400ce32b065c16a557b36dc908be4a3df4ba1697285a769f5f71ca5055d56b5d 4d086d0ec9a2c13fb3fcff25d71345ece3aee12da281ee993db3fb77f48e85f0 751a4132562b28448228715712eddced3d19224ae3155f32ebcab128c4767321 adb9d34f15d2c6e7c41d21c63b7e8b06e2b562199b2b29e1a95ad01aa7aae90a
Decryption Key (string): “tesu”
Cipher Text (binary data):
082317e747f6cbea7b7170a6edac8f6f1a2aac26517ac630a3009689abd83b4d 261d9df081f2d11dcbe77ed991a83fbb25132376f472f5b26c3e48c5280bef1b 12294b25839c7db2470b547c9a18a272d71949d3e2fce3499e8bd46648913075 008391533a60aad01264aaf981e04a6a13fa39bfef6f7f3a57ada2e14fbc99b4 aa70015d533e58709b1e2cfa0a1d07f8c47dfe030a481dcfcb93a989c7cc19c6 015180a121f0e0bacc7a43124d1ff54ac0515b424d8bcdbf5d5d4a1328d74212 b1b8112b6159887eda984ef87aae3c957ddc9cdc319d1bb3cd346d71414d86cf c7a9de85fd3ef62a39514a6c951bb2aed5e14d92ba91708052f9d242e08f5f7e

V. SUMMARY

While this encryption algorithm serves as a proof of concept that replaces simple key scheduling with a more advanced hash-based key scheduling, this encryption algorithm has not been tested thoroughly against real cryptographic attack methods. Furthermore, currently with a test implementation using Java, encrypting a 41KB file requires a time of around 5 seconds. At the current form it is still not suitable for general purpose encryption.

The shuffling of rows and columns can also be improved by using a more sophisticated shuffling method to increase security such as multiplying the columns with a fixed polynomial like in the AES algorithm [3], as well as improving the internal round key scheduling mechanism to something similar to the AES algorithm.

REFERENCES

[1] Claude E. Shannon, "A Mathematical Theory of Cryptography", Bell System Technical Memo MM 45-110-02, Sept. 1, 1945.

[2] Electronic Frontier Foundation, FAQ about the EFF DES Cracker Machine [Online]. Available: https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

[3] Federal Information Processing Standards Publication 197. United States National Institute of Standards and Technology (NIST). "Announcing the ADVANCED ENCRYPTION STANDARD (AES)". November 26, 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[4] Biryukov, Alex; Khovratovich, Dmitry (2009-12-04). "Related-key Cryptanalysis of the Full AES-192 and AES-256". [Online]. Available: https://eprint.iacr.org/2009/317

[5] M. Backes. (2011). Block Ciphers [Online]. Available: http://web.cs.du.edu/~ramki/courses/security/2011Winter/notes/feistelProof.pdf.

[6] William F. Ehrsam, Carl H. W. Meyer, John L. Smith, Walter L. Tuchman, "Message verification and transmission error detection by block chaining", US Patent 4074066, 1976

[7] Schneier, Bruce. "Cryptanalysis of MD5 and SHA: Time for a New Standard". Computerworld. [Online]. Available: https://www.schneier.com/essays/archives/2004/08/cryptanalysis_of_md5.html