A NEW BLOCK CIPHER (NAHRAINFISH) BASED ON SOME AES FINALISTS

SUFYAN T. FARAJ

College of Computers- University of Anbar

ARTICLE INFO

Received: 15 / 3 /2007
Accepted: 20 / 8 /2007
Available online: 14/6/2012
DOI: 10.37652/juaps.2007.15327

Keywords:
AES,
Block ciphers,
DES,
Feistel networks,
Nahrainfish.

ABSTRACT

In this work, we present some deeper insights in the state-of-the-art in block cipher design. This is mainly achieved by assessment of the evaluation process of the AES (Advanced Encryption Standard). We conclude (with may be a little bit surprising result) that the required security level, for a block cipher to be used for the present and foreseeable future, may be underestimated during AES evaluation. In accordance, we propose a new block cipher that we call Nahrainfish, which we believe that it offers the required security level without a big sacrifice in efficiency and other related criteria. Instead of building Nahrainfish totally from scratch, we have made a benefit mainly from some AES finalists to develop the new cipher by an over-engineering approach leading to the required higher security level. Nahrainfish is a classical Feistel network based on a novel combination of both key-dependent S-boxes and data-dependent rotations. It operates on 128-bit blocks and accepts a variable key length up to 1024 bits. The paper also includes some important notes on the security and performance of the cipher.

INTRODUCTION

In general, there are two types of cryptosystems; secret-key (one-key) and public-key (two-key) ciphers. In conventional secret-key ciphers, the same (secret) key can be used in both encryption and decryption. While in public-key systems, two totally different keys (private key and public key) are used one for encryption and the other for decryption. Secret-key ciphers can further be subdivided into two categories; stream ciphers and block ciphers. In a typical stream cipher, a short string of key bits is used to generate a long sequence of bits that is added bitwise modulo 2 to the plaintext to produce the ciphertext. On the other hand, in block ciphers, the plaintext is divided into blocks of a fixed length, which are encrypted into blocks of ciphertext using the secret key.

Block ciphers are the most important elements in many cryptographic systems. Basically, they are used to ensure data confidentiality.

Indeed, they can be used as fundamental building blocks to construct stream ciphers, MACs, hash functions, and pseudorandom number generators. A block cipher may be defined as a pair of two functions; the encryption function (E) and decryption function (D). E turns n-bit plaintext into n-bit ciphertext under control of k-bit key (K). In contrast, D turns n-bit ciphertext into n-bit plaintext under control of K. Thus, n is the block size and k is the key size. Defining Ek and Dk as the respective encryption and decryption functions for each key K, Ek must be invertible for all k, with $E_k^{-1} = D_k$.

Most block ciphers nowadays are "iterated" block ciphers, which consist of the repetition of the same (round) function F for certain number of times R. Nevertheless, each application of F is parameterized by a different "round key". A key scheduling algorithm is used to derive round keys from the master key K. The round function F is usually constructed on

* Corresponding author at: College of Computers- University of Anbar, Iraq.E-mail address: sufyantaih@ieee.org

the basis of the two important principles of Shannon; confusion and diffusion [1]. Informally, confusion aims to make the relation between the statistics of ciphertext and the value of the key as complex as possible. While, diffusion aims to make the statistical relationship, between the bits of plaintext and ciphertext, as complex as possible.

There are two main structures for iterated block ciphers, which are the substitution-permutation network (SPN) and Feistel network. SPNs are block ciphers with a very simple structure. The round of a SPN consists of three layers [2]:

The key mixing (addition) layer $\delta[k]$, which is usually done using bitwise XOR ($\oplus$).

The diffusion layer θ, which is linear with respect to the group operation used in the key addition layer.

The confusion (non-linear) layer γ , which is usually performed by the parallel application of S-boxes.

Hence, the round of a SPN can be described as:

$$\rho[k] = \delta[k] \circ \theta \circ \gamma \qquad (1)$$

where ∘ denotes function compositions.

Usually a key addition layer is added before the first round. Indeed, the last diffusion layer is not cryptographically useful. Hence, the whole R-round SPN can be described as [2]:

$$\delta[k_R] \circ \gamma \circ \left(\bigcirc_{r=1}^{R-1} \delta[k_r] \circ \theta \circ \gamma \right) \circ \delta[k_o] \qquad (2)$$

where k0, k1,…, kR are round keys.

In a (pure) Feistel network, the data X entering a round is divided into two halves (left and right); X=<L,R>. Thus, the round is described as:

$$\phi(f)(<L,R>) = <R, L \oplus f(R)> \qquad (3)$$

where f is the round function, which consists of confusion, diffusion, and key mixing layers.

The main advantage of SPNs is their simple structure, which makes them easy to analyze. On the other hand, the main advantage of Feistel networks is that decryption differs from encryption by the order of using round keys and by initial and final swaps only. Also, the function f needs not to be objective, which allows more freedom in its design [2].

Besides the classical (pure) Feistel structure, it is also possible to address a much wider design space of Unbalanced Feistel Networks (UFNs) or even Generalized Feistel Networks (GFNs). Like conventional Feistel networks, UFNs consist of a series of rounds in which one part of the block (called the "source block") operates on the rest of the block (called the "target block"). The difference is that, in a UFN, the two parts need not be of equal size. A detailed taxonomy of UFNs and GFNs can be found in [3] and [4].

After this introduction, the rest of this paper is organized as follows. Section 2 presents some considerations related to security of block ciphers in general. The AES evaluation process is discussed in Section 3, where our remarks on that process are presented. The remaining sections are dedicated for proposing the new block cipher: Nahrainfish. The design criteria are emphasized in Section 4. Then, various components of the cipher are described in Section 5. While, design motivations are explained in Section 6. Finally, some security and performance issues of Nahrainfish are considered in Section 7, before the paper is concluded in Section 8.

2. SECURITY OF BLOCK CIPHERS

To evaluate block cipher security, it is traditional to always assume that [5], [6]:

1- The attacker has access to all (ciphertext) messages transmitted over the insecure charnel.

2- All keys are equally likely and K is always chosen uniformly random.
3- The attacker knows all details of the encryption and decryption process except the secret key (which security consequently rests entirely upon). This is the so-called kerckoff's assumption.

Under these assumptions, attacks are classified based on what information a cryptanalyst has access to in addition to the intercepted ciphertext. These attack classes are [7]: ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, adaptively chosen-plaintext attack, and chosen-ciphertext attacks. In fact, it is possible to consider any combination of the above attacks. Obviously, the chosen-text attacks are the most powerful ones. However, in many applications, they are unrealistic attacks.

The progress in cryptanalysis is an important factor in the development of new block ciphers. As new cryptanalytic techniques appear, new insights are brought in the security of block ciphers and their basic building blocks such as S-boxes and diffusion layers.

In most modern cryptanalytic attacks on (iterated) block ciphers, the attack procedure is repeated for all possible values of (a subset of) the bits in the last-round key. Thus upon success in guessing the correct values of the key bits, the attacker can compute the ciphertext bits after the second-last round (i.e. before the last round). While a wrong guess means that these bits correspond to ciphertext bits encrypted with a wrong key. Whenever there is a probabilistic correlation between the plaintext bits, and the bits of the ciphertext before the last round, the attacker may be able to distinguish the correct guesses from wrong ones of the last-round key. Thus, the attacker can exclude the last round of the cipher and repeat the attack on the reduced cipher (i.e. a cipher one round shorter) to find the second-last round key, and so on. It is also possible in some cases to consider the first-round key instead of the last-round key or both of them simultaneously. In iterated block ciphers the correlation is usually found by first identifying a correlation between inputs and outputs of individual rounds, and then combining to a correlation over several rounds [7].

There is a variety of cryptanalytic attacks in the literature [7], [8]. Some of the most important attacks are: differential cryptanalysis, linear cryptanalysis, differential-linear attacks, interpolation attacks, non-surjective attack, related-key attacks, slide attacks, multiset attacks, algebraic attacks, and implementation attacks.

3. FROM DES TO AES

For more than two decades, the DES (Data Encrypted Standard) was the dominant and most widely applied block cipher. During this period DES received a lot of scrutiny. However, as technology advances, it was became obvious that a new standard is required for a block cipher that is optimized to today's technology.

Hence, NIST in 1997 published a request for candidates to become a new Advanced Encryption Standard (AES), which has to be at least as secure as 3DES and significantly more efficient. Indeed, the candidates should support 128-bits block size and key sizes of 128, 192, and 256 bits. In a first round of evaluation, 15 submissions were originally accepted. Then, in 1999, NIST selected five of them as finalists: MARS, RC6, Rijndael, Serpent, and Twofish. After three years of a public review process, NIST decided in October 2000 to choose Rijndael (developed by J. Daemen and V. Rijmen from Belgium) as the new AES (FIPS 197).

In the following subsections, each of the five AES finalists is briefly described at first. Then, a summary of the criteria used in the AES evaluation process is given. Finally, some important comments on the AES evaluation are presented.

3.1 AES FINALISTS

There are some common technical features of the five finalist candidates. Four of them (MARS, Rijndael, Serpent, and Twofish) use S-boxes in their design. Three of the finalists (MARS, RC6, and Twofish) specify variations on the classical Feistel structure. While the other two finalists (Rijndael and Serpent) are examples of SPNs. Below is a summary of each of the finalists [9]:

1- MARS:

MARS, which was submitted by IBM, uses a 32-round unbalanced Feistel network: in each round one fourth of the data block is used to alter the other three fourths of the data block. The algorithm has several layers: key addition (pre-whitening), 8 rounds of unkeyed forward mixing, 8 rounds of keyed forward transformation, 8 rounds of keyed backwards transformation, 8 rounds of unkeyed backward mixing, and key subtraction (post-whitening). The 16 middle (keyed) transformations are called the cryptographic core. The designer's intuition is that using top and bottom rounds employing functions different from the middle ones is better resistant to differential and linear cryptanalysis. The unkeyed rounds constitute 8×32 bit S-boxes, 32-bit addition, and XOR operation. While the keyed rounds additionally constitute 32-bit key multiplication, data-dependent rotations, and key addition [10].

2- **RC6**:

RC6 (developed by RSA Laboratories) is an evolutionary extension of an earlier developed cipher RC5 to meet the AES requirements. It is a parameterized family Feistel-structured cipher (20 rounds for the AES submissions). RC6's strength mainly lies in the resistance to differential and linear cryptanalysis provided data-dependent rotations used in the round function. Indeed, each round includes 32-bit modular multiplication, addition, XOR, and key addition. Also, pre- and post-whitening are accomplished using key addition [11].

3- **Rijndael**:

Rijndael, developed by J. Daemen and V. Rijmen, relies more directly on algebraic constructs than other candidates. It is somewhat similar to an earlier cipher (SQUARE) with significant enhancements. Rijndael is a SPN that partition a data block into an array of bytes and uses byte oriented operations. The number of rounds is 10, 12, or 14, depending on the key size. Its round function consists of four layers: a byte substitution layer based on 8×8 S-boxes, a linear mixing layer based on shifting rows of the array, another linear mixing layer using columns mixing (this layer is skipped in the last round), and finally subkey bytes are XORed with each byte of the array in the last layer. This design of the round function facilities many possibilities for parallelism in Rijndael [12].

4- **Serpent**:

Serpent, created by R. Anderson, E. Biham, and L. Knudsen, is a conservative design. It is a 32-round SPN. The round function includes three layers: the key XOR operation, 32 parallel applications of one of the eight specified 4×4 S-boxes, and a linear transformation (the last round replaces the linear transformation with a second layer of key XOR). Its design facilitates bit slice mode implementations. The security of serpent is achieved by a high number of rounds, which makes the cipher to be highly resistant to differential and linear cryptanalysis [13].

5- **Twofish**:

Twofish, proposed by Counterpane Systems, is a 16-round modified Feistel network. Its round function works on 32-bit words with four key-dependent 8×8 S-boxes. This is followed by a fixed 4×4 MDS (Maximum Distance Separable) matrix, a pseudo-Hadmard transformation, and key addition. The designers of Twofish believe dynamically varying key-dependent S-boxes significantly enhance security [14].

3.2 THE AES EVALUATION PROCESS

The AES evaluation criteria originally were divided into three major categories, which are [9]:

1. Security: This is the most important factor in the evaluation and refers to the effort required to cryptanalyze the cipher, with emphasis on practicality of the attack.
2. Cost: This is the second important factor that encompasses algorithm speed on various platforms, memory requirements, and licensing requirements.
3. Algorithm and Implementation characteristics: This category includes other considerations such as flexibility, hardware and software suitability, and simplicity.

With the progress of evaluation process, it was noted that the various issues often crossed into more than one of the three main criteria categories. Thus, the cost and algorithm characteristics were considered together as secondary criteria, after security. Hence, the following criteria were used in the final evaluation:

- General security
- Software implementations
- Restricted-space environments
- Hardware implementations
- Attacks on implementations
- Encryption vs. decryption issues
- Key agility
- Other versatility and flexibility
- Potential for instruction-level parallelism

For the general security criteria, NIST relied on the public security analysis conducted by the cryptographic community. The issues related to this concern that had been considered by NIST in evaluating the AES finalists were:

- Attacks on reduced-round variants.
- Algorithm security margin.
- Design paradigms and ancestry.
- Algorithm simplicity.
- Statistical Testing.
- Other security observations.

Based on the performed security analysis, NIST assessment of the finalists was [9]: "There are no known security attacks on any of the five finalists, and all five algorithms appear to have adequate security for the AES. In terms of security margin, MARS, Serpent, and Twofish appear to have high security margins, while the margins for RC6 and Rijndael appear adequate. Some comments criticized Rijndael for its mathematical structure and Twofish for its key separation; however, those observations have not led to attacks".

At the end of the second stage of the evaluation process, a poll of audience was taken and it showed that Rijndael was public favorite (Rijndael: 89 votes, Serpent: 59, Twofish: 31, RC6: 23, MARS: 13) [8]. On October 2000, NIST selected Rijndael as the proposed AES algorithm. NIST has concluded that [9]: "when considered together, Rijndael's combination of security, performance, efficiency, implementability, and flexibility makes it an appropriate selection for the AES for use in the technology of today and in the future" (Note that based on a similar criteria, Rijndael also was selected as a portfolio of NESSIE project among some other ciphers in 2003 [15]).

3.3 REMARKS ON THE AES EVALUATION

Our main interest in this work is the security criteria for proposing a specific block cipher as a standard for present and foreseeable future applications. No one can under evaluate the work done during the AES evaluation process. However, in such complex projects, there are always trade-offs and design decisions that have to be made based on adopting certain points of view. As there is more than one school of thought in cryptosystem design, we believe that there were some security concerns that have been underestimated during the AES evaluation and selection. In this subsection, these security concerns and issues are discussed.

The first of these issues, which have been reported previously in the literature, is related to the relatively short frame time of the AES evaluation process. Compared to the analysis of DES, the AES evaluation process is quite limited. Even more, one may expect that cryptanalysis effort could be biased towards algorithms that attract greater scrutiny in a limited evaluation timeframe. This is greatly could be affected by the familiarity with certain structures and components that appear to be simpler to be attacked using already known cryptanalysis techniques.

The second issue is concerned with the "qualitative" approach followed by NIST to AES selection. In spite of that the issue of following a rather "quantitative" approach had been raised by the public more than once during the AES evaluation process, NIST felt that such an approach would not be workable. The main reason for that was the degree of subjectivity of many of the criteria [9]. However, we believe that due to NIST selection approach, some important criteria have been underestimated. In contrast, in a quantitative approach, each AES selection factor would be given an explicit weighting, and thus each algorithm would receive a score based on the evaluation criteria. Of course, an agreement on such a quantitative scoring system might require a significant public discussion. Our opinion is that developing a hybrid of both quantitative and qualitative approaches would be even better for such projects.

The third issue is related to the evaluation criteria. NIST stated that security is the foremost concern in evaluating the finalists. And all other concerns were supposed to be secondary criteria. However, as NIST had assessed that "all finalists appear to have adequate security", the choice of a single algorithm (Rijndael) to be the proposed AES was solely made based on the (secondary) criteria of cost and algorithm characteristics. On one hand, this might be the only way to have the work done. On the other hand, this appears as working things up-side-down. Note that while security was one out of three main categories in the original evaluation criteria, it turned to become one out of eight or nine factors in the final criteria (a related security issue was addressed by the category of "Attacks on Implementation"). The crucial question here is that: Do all finalists really have adequate security for the AES? We hope that the answer would be clearer in the remaining of this subsection.

The fourth (and most important) issue is concerned with the category of "General Security" in the final criteria. In spite of that there were several factors considered for security evaluation, it seems that NIST team had no objective criteria except for a resistance to all known cryptanalysis attacks. This was partially due to the "qualitative" approach for selection followed by the team. Since all finalist candidates were built to withstand any known attacks, the role and consequences of other security factors (such as algorithm security margin, design paradigm, and ancestry) were not obvious in the result of security evaluation. Thus, all finalists had been considered to be secure and the evaluation concentrated on performance and flexibility issues as the selection criteria. The criticism on this strategy is that it (implicitly) had assumed that resistance against known attacks is "sufficient" for an algorithm to be secure. While, in fact this is only a "necessary" condition. As far as we are concerned, we believe in an opposite point of view, which can be expressed like this: all the finalists were adequate with respect to speed and flexibility concerns. And the criteria for selection had to be security and "robustness against future (or already known but unpublished) advances in cryptanalysis". It is really difficult to understand why this issue of robustness was not considered in a more sophisticated and serious manner during a process of AES evaluation and selection?

The fifth issue considered here is related to the AES (Rijndael) algorithm itself. This algorithm can be criticized for its (may be insufficient) security margin, its design style (SPN) which did not receive scrutiny like the well-known Feistel networks, and its mathematical structure. In [16], for example, several security concerns about the mathematical structure of Rijndael have been discussed. In 2002, after about two years from selecting Rijndael as the AES, a new cryptanalysis technique appeared: algebraic attacks [17]. If this attack can work as described, then this might represent a non-trivial concern of AES security and robustness. In all cases; when all these issues considered together, one cannot be so optimistic about the actual security of the AES for the proposed life span.

All the issues that are discussed above and many others have been taken into consideration in the design of the new block cipher Nahrainfish, which is described in the rest of this paper.

4. NAHRAINFISH DESIGN CRITERIA

In the second part of this paper, we propose a new block cipher which we call Nahrainfish (Nahrain- refers to the two great rivers of Mesopotamia: Tigris and Euphrates). Nahrainfish works on 128-bits blocks and accept a variable key size (up to 1024 bits). The development of Nahrainfish was done based on the argument on AES evaluation and selection, presented at the first part of this paper. Thus, instead of developing Nahrainfish totally from scratch, its design was mainly inspired by techniques and components used in some AES finalists (especially MARS, Twofish, and RC6) and some other earlier ciphers (e.g., Blowfish [18]).

We adopt the point of view that considers "no block cipher is ideally suited for all applications, even one offering a high level of security" [5]. This point of view is based on the fact that there are various tradeoffs required in practical applications (e.g., due to constraints imposed by implementation platforms, memory limitations, speed requirements, etc). For example, we strongly disagree that the same block cipher used for securing international e-commerce should also be the choice for smart cards. Note that all AES candidates have serious problems related to differential-power analysis in smart cards applications, as reported in [19]. Hence, to avoid system vulnerabilities inherent in symmetric key designs, there is a rapid interest in developing (special-purpose) public-key algorithms for smart cards [20]. Thus, smart card suitability is simply not an issue for Nahrainfish design.

The design criteria for Nahrainfish are as follows:

1. A 128-bit symmetric block cipher.
2. A variable key length (32 bits up to 1024 bits). However, we recommend using keys no less than 256 bits.
3. Resistance against all known attacks.

4. Robustness against future advances in cryptanalysis.
5. Simple, modular, and flexible design in order to facilitate ease of analysis and ease of implementation.
6. Efficiency on 32-bit microprocessor and other important software and hardware platforms.
7. Performance, which is comparable to that of AES finalist in most relevant applications.
8. Trusted cipher, by minimizing the possibility of existence of trapdoors in the cipher.

5. DESCRIPTION OF NAHRAINFISH

Nahrainfish uses a conservative design of 20-round pure Feistel structure with pre-and post-whitening steps, as shown in Figure(1). The plaintext block is divided into four 32-bit words, which are XORed with four key words in the pre-whitening step. Then twenty normal rounds follow. In each of these rounds, the two words on the left are used as input to the round function F. This function also receives two key words. The output of F is two words that used to modify the other two words on the right, using the XOR operation. Then, a swapping of the left and right halves is performed for the next round. After the normal 20 rounds, a reverse swap is done. Finally, post-whitening is accomplished by XORing the resultant four words with additional four key words to produce the ciphertext block.

The following notations are used to describe Nahrainfish:

- a + b integer addition modulo 232.
- a $\oplus$ b bitwise XOR of 32-bit words.
- a $\times$ b integer multiplication modulo 232.
- a <<< b cyclic rotate the 32-bit word a to the left by the amount given by the least significant five (log2 32) bits of b.

5.1 KEY SCHEDULE

The key schedule of Nahrainfish provides 48 32-bit subkey words SK0, …, SK47. It is also used to generate four 8×32 key-dependent S-boxes containing a total of 1024 32-bit entries. The total is 1072 32-bit values (4288 bytes). Nahrainfish accepts a key that ranges from 32 bits to 1024 bits and stores these bits in a K-array, which contains a maximum of 32 32-bit elements:

$$k_0, k_1, \ldots, k_j \qquad 0 \le j \le 31$$

While the subkey 32-bit words are stored in the SK-array:

$$SK_0, SK_1, \ldots, SK_{47}$$

There are also four key-dependent S-boxes, each with 256 32-bit entries:

$$S_{1,0}, S_{1,1}, \ldots, S_{1,255}$$
$$S_{2,0}, S_{2,1}, \ldots, S_{2,255}$$
$$S_{3,0}, S_{3,1}, \ldots, S_{3,255}$$
$$S_{4,0}, S_{4,1}, \ldots, S_{4,255}$$

The process of generating the SK-array and S-boxes proceeds as follows:

1. Initialize first the SK-array and then the four S-boxes in order using the bits of the fractional part of the constant π .
2. Perform a bitwise XOR of the SK-array and K-array, reusing words from the K-array as required.
3. Encrypt the 128-bit block of all zeros using the current SK-array and S-boxes. Then replace SK0, SK1, SK2 and SK3 with the output of encryption.
4. Encrypt the output of step 3 using the current SK-array and S-boxes, and replace SK4, SK5, SK6, and SK7 with the resulting ciphertext.
5. Continue this process to update all elements of the SK-array, and then (in order) all elements of the S-boxes. This to be done using at each step the output of the continuously changing Nahrainfish.

The above steps for subkey and S-boxes generation require a total of 268 Nahrainfish executions. For rapid execution, applications can store the subkeys and S-boxes.

5.2 THE ROUND FUNCTION F

The arguments of the round function F are two input 32-bit words (L0 and L1) and the round number r $(0 \le r \le 19)$ used to select the two appropriate subkey words. F includes two other functions (g and h). In the beginning of F, the two subkey words (SK2r+8 and SK2r+9) are added (modulo 232) to L0 and L1 respectively. Then, L0+SK2r+8 is passed through the g function to yield T0. While, L1+SK2r+9 is passed through the h function to obtain T1. The outputs of the g and h functions (T0 and T1) are combined using a Pseudo-Hadmard Transformation (PHT). A schematic of the round function F is shown in Figure (2).

The PHT is a simple matrix operation that can be defined as follows: let a and b be two 32-bit input words, then the 32-bit PHT is:

a' = a + b (4)

b' = a + 2b (5)

Thus, denoting (F0, F1) to be the output of the function F, F can be more formally described as follows:

$$T_0 = g(L_0 + SK_{2r+8}) \quad (6)$$

$$T_1 = h(L_1 + SK_{2r+9}) \quad (7)$$

$$F_0 = T_0 + T_1 \quad (8)$$

$$F_1 = T_0 + 2T_1 \quad (9)$$

5.3 THE FUNCTION g

The 32-bit input (L0+SK2r+8) to the function g is divided into 4 bytes. Each of these bytes is used as and input to invoke one of the four S-boxes, as shown in Figure (2). The output of the first S-box is combined with output of the second S-box using the XOR operation. Then the result of this step is combined with the output of the third S-box using addition modulo 232. Finally, this result is combined with the output of the fourth S-box using another XOR operation to produce the output T0. If the 4 bytes input to the S-boxes are labeled as a, b, c, and d, then the g function can be defined as follows:

$$T_0 = g(a,b,c,d)$$

$$= ((S_{1,a} \oplus S_{2,b}) + S_{3,c}) \oplus S_{4,d} \quad (10)$$

5.4 THE FUNCTION h

The function h is a data-dependent rotation function that makes use of the quadratic function:

$$h_0(x) = x \times (2x+1) \quad (11)$$

This quadratic function (h0) is applied to the 32-bit input word to the function h, and then a fixed rotation of 5 positions to the left is performed. Finally, the value of the least significant five bits of the result is used to decide the amount of left rotation on the input word, as illustrated in Figure (2). Thus, the function h can be defined as follows:

$$h(L_1 + SK_{2r+9}) = (L_1 + SK_{2r+9}) <<< (h_0(L_1 + SK_{2r+9}) <<< 5) \quad (12)$$

5.5 ENCRYPTION AND DECRYPTION

In order to use Nahrainfish to encrypt a 128-bit (16 bytes) plaintext block, these bytes p0, p1, …, p15 are first split into four 32-bit words P0, …, P3. The little-endian notation is used for this purpose;

$$P_i = \sum_{j=0}^{3} p_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 3 \quad (13)$$

Then, Nahrainfish encryption proceeds according to the pseudo code shown in Figure (3). The result of this procedure is four 32-bit words of ciphertext C0, …, C3. These words are then written as 16 bytes using the little-endian convention;

$$c_i = \left\lfloor \frac{C_{\lfloor i/4 \rfloor}}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8 \qquad i = 0, \ldots, 15 \qquad (14)$$

Since Nahrainfish is a Feistel network, decryption is similar to encryption but using the subkeys in a reverse order.

6. MOTIVATIONS FOR DESIGN CHOICES

In the design of Nahrainfish, we tried to set the highest security and robustness goals, while maintaining a flexible and fast cipher for most relevant applications. Nahrainfish is intended to be used in applications requiring higher security demands for the present and foreseeable future. The principles behind our design were as follows:

1. Security: Design a cipher that is both resistant to all known attacks and robust a gains future advances in cryptanalysis. To obtain a cipher with sufficiently high security margin, it is not enough to rely on a single parameter such as number of rounds (though it is an important parameter). In fact, to achieve this goal, it is necessary to adopt the notion of "minimizing the trust in any single component".
2. Simplicity: It is important to design a cipher that can be analyzed. Ad hoc design elements should not be included without clear reasoning. One should also avoid using "design tricks" whose security consequences are not clear. Indeed, another important concern of the simplicity (of both of the design style and components used) is to produce a cipher that can be trusted to contain no trapdoors (for discussion of trapdoor block ciphers see, for example, [21]).
3. Performance: Within the design principles outlined above, Nahrainfish was designed to achieve high performance and flexibility in most relevant software and hardware implementations.

In accordance to these principles, Nahrainfish was designed as a classical Feistel network, which is based on a novel mixing of key-dependent S-boxes and data-dependent rotations. Note that MARS also uses a mixing of S-boxes and data-dependent rotations. However, MARS is of a different design style, and (more important) MARS S-boxes are generated by a "pseudorandom" process and are not key-dependent.

6.1 THE ROUND STRUCTURE

Nahrainfish was designed as a pure Feistel network, because it is the most well-studied block-cipher structure. We did not choose something newer and/or less-studied structures like UFNs, GFNs, AES (Rijndael)-like, or AES-like structures based on involution components. One important additional advantage of using this structure is that the same algorithm can be used for both encryption and decryption (with reversing the order of subkeys).

6.2 S-BOXES

Nahrainfish uses four 8×32 key-dependent S-boxes that are generated using repeated iterations of the cipher itself. It is expected that such large S-boxes generated this way are of a high non-linearity. We decided not to use other forms of algebraic or tabular S-boxes. This was mainly to give an evidence of minimizing the probability of trapdoor existence, since there cannot be any pre-defined mathematical structure of an S-box. However, the cost of this design decision is a performance penalty in key-setup time. It is also important to note that it was already observed (in [22]) that the efficiency and small computational complexity of some other types of key-dependent S-boxes (such as those used in Twofish) cannot be easily obtained together with the highest level of security.

6.3 The Key Schedule

The style of the key schedule of Nahrainfish is similar to that of Blowfish. Such key schedules are designed so that knowledge of one round subkey does not directly lead to specify bits of other round subkeys. This is done by repeated application of the block cipher algorithm itself to act as a one-way function for generating subkeys (and key-dependent S-boxes). Such key scheduling, besides its high security (given the high security of the algorithm itself) can also be trusted to have no implicit trapdoors (again given that the underlying block cipher algorithm has no trapdoors). In addition, this style facilitates analysis and implementation of the cipher. A disadvantage of Nahrainfish key schedule is that it takes relatively long time for setup, which puts a limit on cipher application. However, this issue can also be viewed as an advantage because it increases the difficulty of key-search brute-force attack.

6.4 THE ROUND FUNCTION

The round function F was designed to include two sources of nonlinearity: the key-dependent S-boxes in function g and the data-dependent rotation in function h. The major mechanism for providing diffusion in F is using PHT, which combines the outputs of functions g and h so that both of them will affect both 32-bit target words. PHT had been used previously in other ciphers such as Twofish. It facilitates very fast operation on Pentium processors.

The function g is similar to the round function in Blowfish. However, this function has been re-tailored for Nahrainfish by interchanging the locations of XOR and addition (modulo 232) operations. Since these two operations do not commute, we expect this interchange to enhance the security of Nahrainfish.

The function h was designed to exploit operations (such as rotations and 32-bit integer multiplication), which are efficiently implemented on modern processors. Integer multiplication was used to compute data-dependent rotation amounts so that the rotation amounts are dependent on all of the bits of the operand.

To achieve this goal, the function h0(x)=x×(2x+1) (mod 232) was used. Note that the same function was also used in RC6 for the same purpose. The most significant bits of h0(x) are the "stronger bits" since they are affected by almost all the input bits. Thus, it is traditional to use a fixed rotation (by 5 positions) after h0(x) so that the five highest bits of the product become the five lowest bits. Then, these bits are used to determine the rotation amount used. Indeed, this technique enhances the diffusion of the cipher. Nahrainfish is novel in that the amount of data-dependent rotation is derived heavily from all the bits of the same word to be rotated.

7. NAHRAINFISH SECURITY AND PERFORMANCE

The block size and key length of Nahrainfish were chosen to be compatible with AES requirements. About ten years ago, it was estimated (in [23]) that with respect to an exhaustive key search, a key size of at least 90 bits would be sufficient for the next 20 years. NIST specified key lengths of 128, 192, and 256 bits for the AES. However, to maintain higher security margins for Nahrainfish, we recommend using keys of at least 256 bits length. Concerning the 128-bit block size, it is widely accepted now that such block size is a good choice with respect to both security and efficiency requirements.

With a block size of 128 bits, a dictionary attack on Nahrainfish (or any other block cipher with the same block size) will require 2128 different plaintexts so that an attacker encrypt or decrypt arbitrary messages under an unknown key. Another limit imposed on Nahrainfish by using a block size of 128 bits, is that it is expected to find differentials with probability 2-120. In theory, such differentials can be found by exhaustive search. This also applies to all block cipher (with similar block size) as their probabilities are only affected by the block size [13]. When using 128-bit, 192-bit, and 256-bit keys, the complexity of key search attack will be 2128, 2192, and 2256, respectively. However, in order to avoid collision attacks (such as those in [24] and [25]), it is prudent to change keys well before 264 (2n/2) blocks have been encrypted.

In recent years, there has been a research trend to construct ciphers which were proven to be secure against known attacks, such as differential, linear, and related key cryptanalysis. However, it is important to remember that provable security against one or more important attacks does not imply that the cipher is secure. This is simply because other attack scenarios might exist. In accordance, we did not try to optimize Nahrainfish against known attacks. Instead, a conservative design approach and over-engineering techniques have been followed in order to make Nahrainfish strong against both known and unknown attacks.

An important security feature of Nahrainfish is that no two consecutive operations use the same structure. In other words, the ordering of operations in Nahrainfish alternates among different (non-commutating) groups. This would make it very difficult for an attacker to exploit a single algebraic structure for launching an attack.

Nahrainfish uses a key schedule that is similar to that used in Blowfish (i.e., the repeated iteration of the cipher itself). It was reported earlier in [26] that Blowfish has weak keys that generate bad S-boxes (the odds of getting them randomly are 1 in 214). This can enable an attack against reduced-round variants of Blowfish. It is; however, completely ineffective against 16-round Blowfish. Accordingly, Nahrainfish has been designed with additional important source of nonlinearity: the data-dependent rotations (the only source of nonlinearity in Blowfish is the key-dependent S-boxes). Thus, we believe that such an attack would not be applicable to Nahrainfish, because it cannot exploit the existence of weak keys (if they exist) in the key schedule.

It is also worth to mention that there are some previously reported (theoretical) attacks on some implementations of data-dependent rotations (e.g., in the RC5 cipher). These attacks are based on the fact that the "rotation amounts" do not depend on all the bits of the operand. In all cases, these attacks cannot be applied to Nahrainfish because the amounts of the used data-dependent rotations are calculated based on all the bits of the operand (this is also true for RC6 and MARS).

Due to the high nonlinearity and complexity resultant from the novel combination of key-dependent S-boxes and data-dependent rotations in Nahrainfish, we expect that the interpolation attack cannot be applied on full 20-rounds Nahrainfish. Moreover, attacks of differential and linear cryptanalysis can only be feasible on small-round versions of Nahrainfish. We believe that 16 rounds of Nahrainfish are adequate for security. However, we choose Nahrainfish to be of 20 rounds to achieve a sufficiently high security margin for long life span. We conjecture that to attack Nahrainfish the best approach available to the cryptanalyst is that of exhaustive key search.

Despite the conservative design and high security margin of Nahrainfish, the algorithm still achieves a comparable performance to AES finalists in software (see Table. 1). Our current (not fullyoptimized) C implementation of Nahrainfish achieves encryption speed of about 66.8 Mbit/sec on 866 MHz Intel Pentium III processor with 128 MB RAM. The code was developed using Microsoft Visual C/C++ V.6 environment with computer running Windows 2000 Professional. Using key length of 256 bits for encryption, many tests have been done, and the results in Table. 1 are the average of these runs. It is expected that an optimized revision of the programming code will result in a significant improvement of Nahrainfish performance. This expectance is to be verified in the near future.

8. CONCLUSION

It is important to reach to a better understanding of the requirements and considerations for a block cipher to be used in the present and foreseeable future applications. We have tried to do so by careful study and analysis of the AES evaluation process. Accordingly, we have proposed a new 128-bit block cipher Nahrainfish based on the well-studied Feistel structure. Nahrainfish has been over-engineered to be resistant to known attacks and robust to future advance in cryptanalysis. Indeed, our design strategy gives a high level of confidence that no trapdoors have been inserted. We could not find significant weakness. However, it is highly interested to see detailed cryptanalysis results obtained by others. Moreover, Discussion of various aspects of implementing Nahrainfish on different hardware and software platforms are also welcomed.

Table. 1: Comparison of encryption speed of different algorithms (with key length of 256 bits).

Algorithm	Encryption Speed (Mbit/sec)
RC6	198.3
MARS	105.5
Twofish	92.4
Rijndael	80.8
Nahrainfish	66.8
Serpent	54.2

Acknowledgments:

The author would like to thank the students of his M.Sc. "Internet Security" course for their big help in C code development, for running many of the tests, and for useful discussions.

REFERENCES:

[1] Shannon, C.E. (1949). Communication theory of secrecy systems. Bell System Technical Journal, 28: 4, pp. 656-715.

[2] Piret, G-F (2005). Block Ciphers: Security Proofs, Cryptanalysis, Design, and Fault Attacks, Ph.D. Thesis, Universite' Catholique de Louvain (UCL).

[3] Schneier, B. and Kelsey, J. (1996). Unbalanced Feistel networks and block cipher design. FSE'96, LNCS 1039, Springer-Verlag, pp. 121-144.

[4] Nyberg, K. (1996). Generalized Feistel networks. Advances in Cryptology-ASIACRYPT'96, LNCS 1163, Springer-Verlag, pp. 91-104.

[5] Menezes, A. et al (1997). Handbook of Applied Cryptography. CRC Press, Inc.

[6] Schneier, B. (1996). Applied Cryptography. John Wiley & Sons, Inc.

[7] Knudsen, L. (1999). Contemporary block ciphers. LNCS 1561, Springer-Verlag, pp. 105-126.

[8] Biryukov, A. (2004). Block ciphers and stream ciphers: The state of the art. Katholieke Universiteit Leuven (KUL), Belgium, (crypto-eprint).

[9] Nechvatal, J. et al (2000). Report on the Development of the Advanced Encryption Standard (AES). NIST, USA.

[10] Burwick, C. et al (1999). MARS - A Candidate Cipher for AES. AES algorithm submission, USA. Available at http://www.nist.gov/aes

[11] Rivest, R. et al (1998). The RC6TM Block Cipher. AES algorithm submission, USA. Available at http://www.nist.gov/aes

[12] Daemen, J. and Rijmen, V. (1999). AES Proposal: Rijndael. AES algorithm submission, USA. Available at http://www.nist.gov/aes

[13] Anderson, R. et al (1998). Serpent: A proposal for the Advanced Encryption Standard. AES algorithm submission, USA. Available at http://www.nist.gov/aes

[14] Schneier, B. et al (1998). Twofish: A 128-Bit Block Cipher. AES algorithm submission, USA. Available at http://www.nist.gov/aes

[15] NESSIE Project (2003). New European Schemes for Signatures, Integrity and Encryption, EU. Available at http://cryptonessie.org

[16] Schroeppl, R. (2000). E-mail comment, AES Round2 public comments, May 15. Available at http://www.nist.gov/aes

[17] Courtois, N. and Pieprzyk, J. (2002). Cryptanalysis of block ciphers with overdefined systems of equations. Advances in Cryptology-ASIACRYPT 2002, LNCS 2501, Springer-Verlag, pp. 267-287.

[18] Schneier, B. (1994). Description of a new variable-length key, 64-bit block cipher (Blowfish). FSE'93, LNCS 809, Springer-Verlag, pp. 191-204.

[19] Chari, S. et al (1999). A cautionary note regarding evaluation of AES candidates on smart cards. 2nd AES Conference, Italy.

[20] IBM MARS Team (2000). MARS and the AES selection criteria. AES public comment, May 15. Available at http://www.nist.gov/aes

[21] Rijmen, V. and Preneel, B. (1997). A family of trapdoor ciphers. FSE'97, LNCS 1267, Springer-Verlag, pp. 139-148.

[22] Macchetti, M. (2002). Characteristics of key-dependent S-boxes: The case of Twofish. Politecnico di Milano, Milan, Italy, (crypto-eprint).

[23] Blaze, M. (1996). Minimal key lengths for symmetric ciphers to provide adequate commercial security. A report by an ad hoc group of cryptographers and computer scientists, USA.

[24] Biham, E. (1996). How to forge DES-encrypted messages in 228 steps. Technical Report CS884, Technion.

[25] Oorschot, P. and Wiener, M. (1994). Parallel collision search with application to hash functions and discrete logarithms. Proceedings of the 2nd ACM Conference on Computer and Communications Security, pp. 210-218.

[26] Vaudenay, S. (1996). On the weak keys in Blowfish. FSE'96, LNCS 1039, Spinger-Verlag, pp. 27-32.

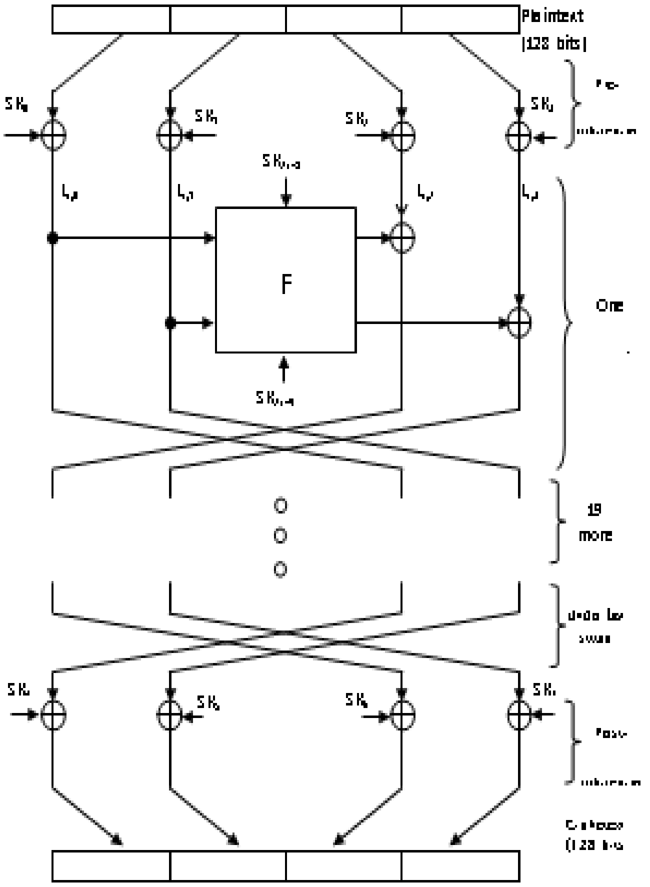


Figure (1): *Nahrainfish Overview*

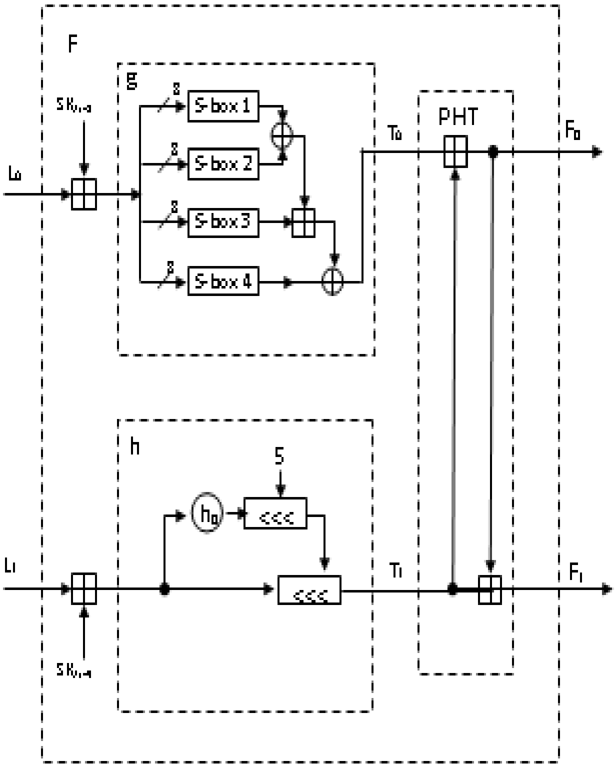
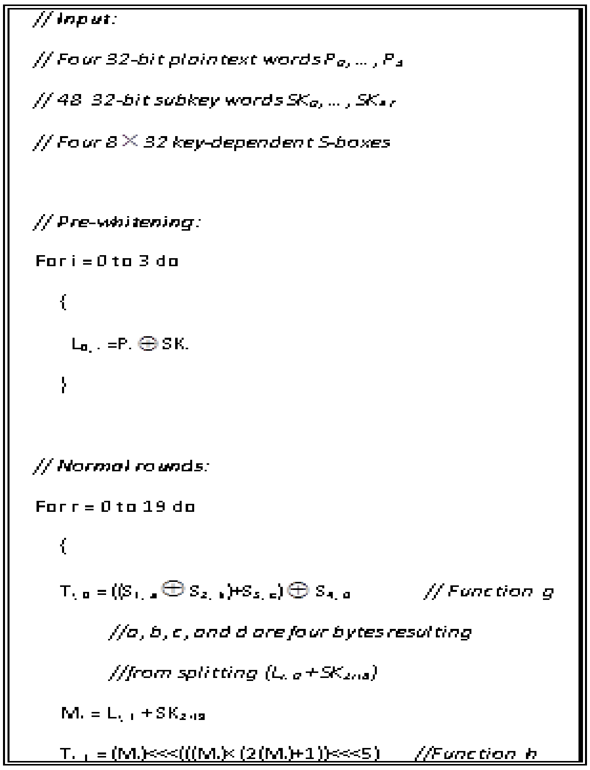


Figure (2): The round function F.
($h_0(x)=x\times(2x+1)$)



```
// Input:
// Four 32-bit plaintext words P0, ... , P3
// 48 32-bit subkey words SK0, ... , SK47
// Four 8 × 32 key-dependent S-boxes

// Pre-whitening:
For i = 0 to 3 do
    {
      $L_{0,i} = P_i \oplus SK_i$
    }

// Normal rounds:
For r = 0 to 19 do
    {
    $T_{r,0} = ((S_{1,a} \oplus S_{2,b}) + S_{3,c}) \oplus S_{4,d}$        // Function g
            //a, b, c, and d are four bytes resulting
            //from splitting ($L_{r,0}+SK_{2r+8}$)
    $M_r = L_{r,1} + SK_{2r+9}$
    $T_{r,1} = (M_r) <<< (((M_r) \times (2(M_r)+1)) <<< 5)$      //Function h
```

Figure (3): Nahrainfish encryption pseudocode

الشفرة المقطعية الجديدة "سمكة النهرين" المعتمدة على بعض الخيارات النهائية لنظام التشفير القياسي المتقدم

د.سفيان تايه فرج

جامعة الانبار – كلية الحاسوب
E.mail: sufyantaih@ieee.org

الخلاصة:

يتم في هذا البحث تقديم بعض الملامح العميقة للمستلزمات الحالية لتصميم الشفرات المقطعية؛ وذلك وبصورة رئيسية من خلال محاولة تقييم عملية اختيار النظام القياسي للتشفير المتقدم (AES). وسيكون الاستنتاج هنا انه ربما لم يتم اعطاء الحجم الحقيقي لبعض الضوابط والمتطلبات الأمنية للاستخدامات الحالية والمستقبلية للشفر المقطعية خلال عملية الاختيار تلك. وبناء على ذلك نقوم باقتراح شفرة مقطعية جديدة نسميها "سمكة النهرين"، والتي نعتقد انها توفر المستوى الأمني المفترض بدون التضحية بالكفاءة والمعايير الأخرى.

إن شفرة "سمكة النهرين" لم يتم بناؤها من الصفر تماما، إنما تم الاستفادة من بعض التقنيات المستخدمة في بناء بعض الشفر النهائية التي تنافست في عملية اختيار النظام القياسي للتشفير المتقدم، وذلك من خلال منهج تصميم هندسي يقود إلى المستوى الأعلى المطلوب للأمنية. وشفرة "سمكة النهرين" تمثل شبكة "فيستل" نمطية تقوم على تركيبة مبتكرة من صناديق التعويض التي تعتمد على مفاتيح التشفير، ومن التدويرات المعتمدة على قيمة البيانات. وتعمل هذه الشفرة على مقاطع بحجم 128 بت، وتقبل مفتاح تشفير ذا طول متغير وحتى حجم 1024 بت. كما يتم في هذه الورقة طرح بعض الملاحظات المهمة حول أمنية وأداء الشفرة المقترحة.