Developing a new Hybrid Cipher using AES, RC4 and SERPENT for Encryption and Decryption

Naser Aghajanzadeh
Islamic Azad University, Qazvin Branch, Computer &Power Faculty, Qazvin, Iran

Fatemeh Aghajanzadeh
Islamic Azad University, shar Rey Branch, Technical & Engineering Faculty, Tehran, Iran

Hamid Reza Kargar
Islamic Azad University, Qazvin Branch, Computer &Power Faculty, Qazvin, Iran

ABSTRACT

This paper aims at developing a new hybrid cipher by combining the characteristics of 3 ciphers namely AES (Advanced Encryption Standard),Rc4 (also known as ARC4) and Serpent. The characteristics of both the ciphers are studied and a new cipher combining the characteristics of both the ciphers is generated which is more secure than the original ciphers. AES, SERPENT characteristics are its security and its resistance against attacks and the major characteristic of Rc4 is its speed.

Therefore these characteristics are imbibed in the newly generated cipher. Thus it proves to be faster than the original AES and secure against most attacks. Three combination techniques have been formulated to generate a hybridized cipher and the procedure along with the strengths and weaknesses outlined. The third cipher is the major cipher which is focused on in this paper. It is also shown that this cipher is resistant against most attacks. This will ensure the secrecy and confidentially of the messages it is used to encrypt.

General Terms

Diffusion analysis, Hybrid, speed improvement, combination of ciphers, confidentiality, encryption, decryption

Keywords

Encrypt, speed, security, AES, Rc4, SERPENT, hybrid, confidentiality.

1. INTRODUCTION

Encryption is the process of transforming plaintext data into ciphertext in order to conceal its meaning and so preventing any unauthorized recipient from retrieving the original data.

Hence, encryption is mainly used to ensure secrecy. Companies usually encrypt their data before transmission to ensure that the data is secure during transit. The encrypted data is sent over the public network and is decrypted by the intended recipient. Encryption works by running the data (represented as numbers) through a special encryption formula (called a key)[1].

If it is possible to determine the plaintext without knowing the key, it can be concluded that the cipher is insecure and easily broken. A cipher attack or a broken cipher means that a third party ends up getting the information. The science used in cipher attack to recover any data or to forge the data as to try to pass it off as authentic is known as cryptanalysis[2].

Data is most vulnerable during its transmission across network and this is where cryptography plays a major role by protecting data through encryption. If the cost or time required to break a cipher is too high to be pragmatic, then the cipher is said to be computationally secure. If a cipher is secure even against attackers with unlimited time and resources then the cipher is said to be unconditionally secure. If a cipher is able to achieve even computational security then it is a secure cipher and can be used in many applications. Unconditional security is not feasible which is why the aim is to strive for computational security[1].

2. BACKGROUND AND RELATED WORK

Various studies and research has been conducted to analyze and study the comparative performance of various algorithms. [3] Analyses various symmetric encryption techniques and compares them on points such as avalanche effect by varying key or plaintext. A study on the combination of Rc4 and AES has been constructed and many probable theories have been outlined for the development of a new algorithm. [4] Formulates a theory on combining block and stream ciphers to give a more complex hybrid cipher. [5] Studies the various attacks on Rc4 and concludes that Rc4 is more secure if hash functions are used in the formation of session keys. Rc4 encryption decryption speed depends mainly on the key length and size of the data provided [6]. Data type is important as it takes more time to encrypt images compared to encrypt text [6]. There have been attempts to improve the security of Rc4 algorithm by using other ciphers or other well-known secure techniques. One such attempt is to combine Rc4 with the polyalphabeticVigenèrecipher to produce a more secure cipher [7]. Cryptanalysis of Rc4 like ciphers [8] finds that the key stream of Rc4 can be tracked and the parts of a key can be recovered given a smaller key size.

3. OVERVIEW OF THE ALGORITHMS

3.1 AES

It is a symmetric-key algorithm (same key is used in both encryption and decryption) and based substitution-permutation design that makes AES so secure against attacks. It is a block cipher which means it breaks data in blocks and combines key with each to get encrypted data[9].

AES has transformation rounds which are called a definite number of times to encrypt data depending on the bit length

Of the key used in the algorithm i.e. 10, 12 or 14 rounds are used for 128, 192 or 265 bit key respectively. The rounds can be called in the reverse manner to decrypt the ciphertext[10].

```
AES pseudo-code:
Function modifiedAES
Pass in: input[], key
State ← Input
Generate Round Key
Add Round Key
For round →1 to round→5
{
Mix Columns
AddRoundKey
}
Sub Bytes
Shift Rows
Add Round Key
Output ← state
Pass out: output []
                End
```

Sub Bytes Function

Shift Rows

As can be seen, the final round will differ only in the aspect that it will not have the Mix-Columns transformation.

3.1.1 Transformation Rounds:

1. Substitute Bytes: A substitution box is created referring which one byte can be replaced with another[10].

2. Shift Rows: Using a certain pre-defined formula each row is shifted a few steps in a cyclic manner[10].

3. Mix Columns: This combines the 4 bytes in each column.

4. Add Round Key: Round key is added again so that the procedure can be repeated for the next round[9].

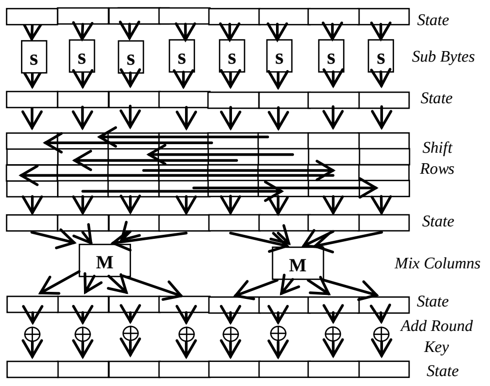


Figure 1: Transformations for one round of AES

3.1.2 Advantages:

This cipher is highly secure, shows a high performance and is fast in both hardware and software because of its substitution permutation network. It has low memory requirements and is easiest to defend against power and timing attacks[11].

3.1.3 Disadvantages:

It is relatively new, complex and doesn't have many implementations till date. It is not very popularly used in WEP because of hardware limitations since it is not as fast in hardware as Rc4. Another vulnerability is that it is a block cipher. And if the same key or derived key is used with more than one block it makes the process of breaking the other blocks relatively easier[9].

3.2 Serpent

Serpent ([12]) is a symmetric block cipher that belongs to a class of substitution-permutation networks (SPN). It was developed by Ross Anderson (University of Cambridge Computer Laboratory), Eli Biham (Technion Israeli Institute Of Technology), and Lars Knudsen (University of Bergen, Norway). In the version that was submitted for AES contest the method operates on 128 bit blocks of data using in the processes a 256 bit external key. The transformation flow is divided into 32 uniform rounds repeated over the data block with each round consisting of (nearly identical) sequence of elementary operations. Each round requires its special 128-bit round key; since the last round needs two keys, total of 33 different round keys are required and these are generated from the external key in a separate key schedule[13].

Fig. 2 represents data transformations that constitute the encryption process. Let *P* be a 128b plaintext, *Bi* – a data Block that enters the *i*-th round *Ri*, *Ki* – the round key, *C* – encoded ciphertext. Before the plaintext block enters the procedure a special bit reordering – so called InitialThen the Final Permutation *FP* (which is an inverse of *IP*) is applied to give the ciphertext*C*. Inside the 32 rounds the actual encoding is carried out.

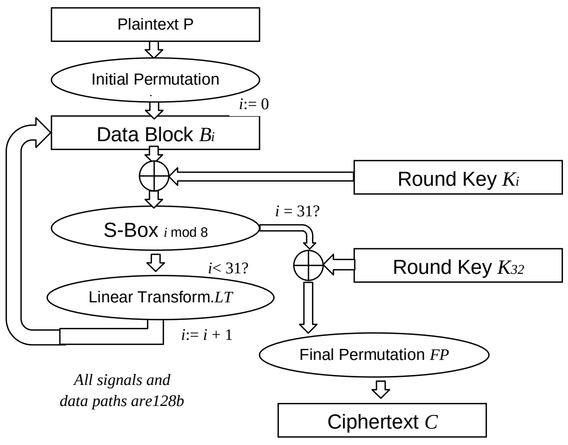


Figure 2: data transformations in Serpent

The first 31 ones (0…30) are identical and the last one (31) is slightly modified. As the first transformation in each round, the block *Bi* is XOR-ed with the round key *Ki* that is supplied by the key schedule. The resulting vector is then passed through *Substitution Boxes*. The algorithm defines 8 different S-Boxes numbered 0 … 7 with each round *Ri*using S-Box number *i* mod 8. The vector created by S-Boxes undergoes *Linear Transformation LT* giving block *Bi*+1 that is the input to the next round. In the last round *R*31 the linear transformationis replaced with XOR operation with the last key *K*32 (therefore two keys are required in this round). The whole data path from the plaintext *P* to the ciphertext*C* can be formally described by a sequence of the following equations:

$$B_0 := IP(P) \quad (1)$$
$$B_{i+1} := LT(Sbox_{i \bmod 8}(B_i \oplus K_i)),\ i = 0 \ldots 30 \quad (2)$$
$$B_{32} := SBox_7(B_{31} \oplus K_{31}) \oplus K_{32} \quad (3)$$
$$C := FP(B_{32})$$

B. Elementary Transformations

The three elementary operations that make up the rounds are: key mixing, bit substitution and linear transformation. Of these three, the first one is just an 128-bit 2-input XOR operation, but the latter are more evolved and require more complex implementations that must be designed according to specification[13].

Like in Rijndael, static substitution boxes perform the nonlinear transformation of the data block in every round. Unlike the AES winner which applies repeatedly the same one 8x8 substitution, the Serpent defines 8 different 4x4 S-boxes (i.e. mappings of 4 bits into 4 bits) with each round using just one S-box. As a result each S-box is used in precisely four rounds, and in each of these it is used 32 times in parallel to transform the whole 128-bit block. In the initial version of the algorithm the authors adopted the S-boxes from DES in order to ensure a high level of public confidence that no secret trapdoor was inserted in them. Later, after public investigation of properties of DES S-boxes that was inspired by new advances in differential and linear cryptanalysis, a new (and better) ones were proposed with even stronger immunity to attacks. Again, to keep high level of public confidence their contents wasgenerated by a special numerical routine which was explicitly clarified and justified. As far as the linear transformation that concludes the rounds is concerned, initially simple rotations of the 32-bitsubwords were proposed. In order to ensure maximal avalanche effect, the idea was to choose these rotations in away that guaranteed maximal effect in the fewest number of rounds. However, as the avalanche was still slow, the authors had to move to more complex transformations and found the XOR operation sufficiently effective: each output bit of the *LT* is the exclusive-or of specific (from 3 to 7) input bits. More complex operations like words addition were also investigated but their cost was too high in both hardware and software implementations and therefore they were dropped ([12]).

3.2.1 The Key Schedule (Fig.3)

The task of the key schedule is to generate 33 round keys *Ki* from the external key *K* that is supplied by the user. The key *K* can be of almost any length but when the proposal for AES standard was formulated it was fixed at 128, 192 or 256 bits with special expansion procedure that is to be applied to keys

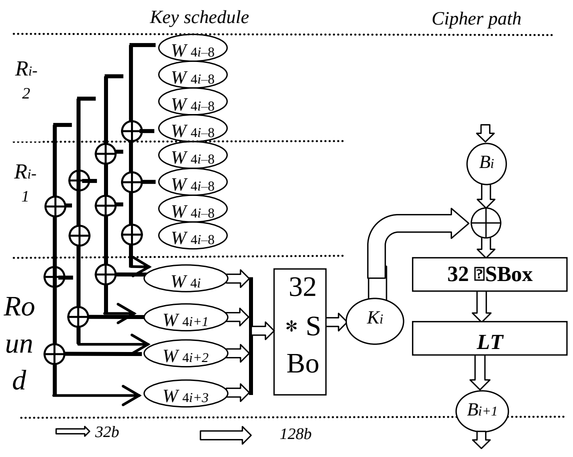


Figure 3: Key Schedule in Serpent

with less than 256 bits. The procedure maps short keys to the full length by appending one "1" bit to the MSB end, followed by as many "0" bits as required to make up 256 bits. This routine maps every short key to a full-length key with no two short keys being equivalent and ensures that the key schedule receives as an input external key that is exactly 256 bits long. The schedule first creates a set of 32-bit *prekeyswi*. The starting 8 prekeys numbered from –1 to –8 are simply filled with bits of the external (user) key *K*:

$\{w_{-1}, w_{-2}, \ldots w_{-8}\} := K$ (5)

and then another 132 prekeys$w_0 \ldots w_{131}$ are generated by the following affine recurrence:

$w_i := (w_{i-1} \oplus w_{i-3} \oplus w_{i-5} \oplus w_{i-8} \oplus \phi \oplus i) <<< 11$

where <<< denotes rotation and $\oplus$ is the fractional part of the golden ratio$(\sqrt{5}+1)/2$(represented as 32-bit vector0x9E3779B9in hexadecimal notation).The underlyingpolynomial $x^8 + x^7 + x^5 + x^3 + 1$ is primitive, which togetherwith the addition of the round index guarantees an even distribution of key bits throughout the rounds and eliminates weak and related keys.The round keys are now calculated from the prekeys using the same set of 8 substitution boxes that are defined for the cipher path. The general rule is that the key *Ki* is computed from a group of four prekeysw_{4i}, w_{4i+1}, w_{4i+2} and w_{4i+3} that undergoes bit substitution and reordering:

$K_0 := IP(\, SBox_3(\, w_0, w_1, w_2, w_3\,)\,)$
$K_1 := IP(\, SBox_2(\, w_4, w_5, w_6, w_7\,)\,)$
… (7)
$K_{31} := IP(\, SBox_4(\, w_{124}, w_{125}, w_{126}, w_{127}\,)$
$K_{32} := IP(\, SBox_3(\, w_{128}, w_{129}, w_{130}, w_{131}\,)$

To avoid repetitive use of the same substitution as later in the round, during computation of *Ki* the schedule uses S-boxes number $(3 – i) \bmod 8$[14,15,16].

3.3 Rc4:

It is a symmetric key cipher like AES and Serpent . It is a stream cipher which means that the random key generated in Rc4 is applied to each bit of the plaintext one at a time to get the encryptedtext.

```
Rc4 pseudo-code:

//Initialization
For i →1 to i→255
Array S[i]← i
Array T[i]←array K[K mod length]

//permuting S
I←0
For j→0 to j→255
I← sum (I, S[j], T[j]) mod 256
Swap S[j] and S[i]

//generating stream
I,j←0
While k<input_length
I←sum(i,1)mod 256
J←sum(j,S[i])mod256
Swap(S[i],S[j])
Temp←sum(S[i],S[j])mod256
K←S[temp]
Output←Input XOR k
End while
```

Here K is the key and S is a state vector which contains a permutation of 8 bit numbers in the range 0 to 255 and one of these numbers is chosen according to predefined rules for encryption and decryption[17].

Here, length denotes the length of the key used. If length is 255 then the key is copied directly to array T else the key is copied and for the remaining part of T, the procedure is repeated continuously till array T has the required number of elements.

Using the key values in the array T, the values in the array S are permuted. Once the initial S vector is ready, keep swapping the values in S according to values in the current array of S. This process is repeated continuously. The S generated is used to obtain the value of byte k. The k can be XORed with the input to encrypt or decrypt the data.

It uses the logic that (A XOR B) XOR B = A, where B=key.

3.3.1 Advantages:

It is very fast, popular and trademarked. It is a simple, easy to implement algorithm. The output or the ciphertext generated by using Rc4 algorithm looks completely random and randomization is a key factor in making a cipher strong.

3.3.2 Disadvantages:

It is a stream cipher, uses a linear operation which can be easily reversed thus prone to attacks and not very secure. It is not recommended in newer applications which require more security. A serious issue regarding Rc4 is discussed in [9], a large number of weak keys were identified and the knowledge of a few bits off the keys could reveal the output[18,19].

4. EXPERIMENTAL DESIGN

The designs proposed are better than the original algorithms. The first two are better in terms of security and the 3rd is better in terms of security as well as speed. Rc4 which is a simple stream cipher uses simple invertible operations. If the invertible operations used are linear then the cipher can be easily broken [4]. Ex: XOR operation.

Take the following example: If P is the plaintext and X1 is the key used then ciphertext Ccan be generated as follows:$C=P\oplus X1$

To get back the plaintext, the ciphertext needs to beXORed with the key again.$C\oplus X1=P\oplus X1\oplus X1=P$

ie. $C\oplus X1=P$

If the plaintext is known the key X1 can be easily generated.

$C\oplus P=X1$

Another case would be:
If P1 and P2 are two different plaintexts and the same key X1 is used to generate the cipher text,
$C1=P1\oplus X1$
$C2=P2\oplus X1$
To generate the plaintexts,
$C1\oplus X1=P1$
$C2\oplus X1=P2$
Since the key used is same,
$C1\oplus P1=C2\oplus P2$
The above equation shows that if P1 is known, P2 can be generated and vice-versa. Even if P1 and P2 are both unknown, most of the plaintext can be easily broken by cryptanalysis. Hence, using the same key with a linear operation for different plaintexts is a liability.
In order to improve security, if 2 keys are used instead of one for the same plaintext it would increase the overhead of generating 2 keys and it would not even make the cipher secure enough. Eg. Assume P for plaintext, C for ciphertextand X1 and X2 as the 2 keys used to generate the ciphertext. If linear operation is used to generate the cipher text then:$C=P\oplus X1\oplus X2$$X1\oplus X2$ is same as $X2\oplus X1$. So even if 2 keys are used in different order, the result will end up being the same as using a single key which is same for 2 different plaintexts.To get back plain text all that needs to be done is:
$P=C\oplus X2\oplus X1$$P1=C1\oplus X2\oplus X1$$P2=C2\oplus X2\oplus X1$

Again, since the keys used in both the plaintexts are equivalent it implies$C1\oplus P1=C2\oplus P2$.
Thus knowing one variable will mean that the other can be easily known which concludes that linear operation does not provide much security in terms of possible attacks.
A solution for this is to use a different key each time which is not pragmatic unless a record is maintained of all the used keys. But a compensation is that the period of rc4 key generation is greater than 10^{100} [16], so the keys repeat after a long period which is quite impractical to trace.
Another solution is to use a nonlinear operation but one which is invertible so that decryption is also possible. Eg.Addition or subtraction.
If X1 and X2 are the 2 keys used and P is the plaintext then

$$C=P\oplus X1+X2$$

In order to get back the plaintext,

$$(C-X2)\oplus X1=P$$

This would be much more difficult to break. Decryption will be easy if both X1 and X2 are known, but without the knowledge of even one of them, it is difficult to break. Even if same keys are used with two different plaintexts, this would be difficult to break.

$$(C1-X2)\oplus X1=P1$$
$$(C2-X2)\oplus X1=P2$$

Since XOR is not distributive, it is not possible to get an equation having only C and P due to which non-linear operations are much stronger.
But even this is susceptible to attacks. Hence combining it with an algorithm like AES would make it more secure. In AES, the transformations are called again and again and the plaintext is shuffled continuously to generate a random looking ciphertext. By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys [15].

5. NEED FOR HYBRID CIPHER

With the growing trend of using computers and internet for all purposes, sending data securely has become highly risky. Hence, security is the growing need of the day which stream ciphers like Rc4 are unable to provide.
WEP application uses Rc4 but Scott Fluhrer in his paper Weaknesses in the Key Scheduling Algorithm of RC4 [12] throws light on the risk factor. He shows how knowing a few bits of the key in the Rc4 cipher can easily break the cipher and determine the output of the cipher with a high probability. It was shown that for a ciphertext attack, a key of arbitrary length could be easily recovered using this technique which renders the cipher highly insecure.
AES has been suggested as a replacement on several occasions but AES being new and a block cipher, it is not as popular as Rc4. Moreover, AES is very slow compared to Rc4 which is one of the fastest ciphers known and is the major reason for its popularity.
As security issues continue to arise, it is time to look at an alternate approach which is why the proposed algorithm can prove to be a cross between Rc4 and AES combining the characteristics of time and speed into a new cipher.
Rc4 combined with AES is highly likely to create a secure algorithm. RC4 can be combined with AES in various ways.

5.1 Rc4Aes

(i)Rc4 output is given to AES as an input. Thus the ciphertext generated by Rc4 is the plaintext for AES and the required ciphertext is the output produced by AES. To decrypt the ciphertext reverse the procedures i.e. use AES decryption 1st and give the output to Rc4 to decrypt and generate the initial plaintext.
The advantage here is that even if AES is broken, the plaintext cannot be recovered as there is another layer of security in the form of Rc4.
(ii)Rc4 produces a random looking ciphertext as its output which becomes a key in AES operations. Here, the plaintext is what is given as a plaintext in AES and the ciphertext is AES output. For decryption reverse the process. Use AES decryption with the Rc4 output as the key. The output generated will be the initial plaintext.
The advantage here is that even if AES is broken, the key remains unknown as the pseudo key (the output of Rc4) is attacked and the actual key remains safe.

5.2 Rc4-Serpent

The output of Rc4 is used as plaintext in Serpent in a similar way as done in Rc4AES.
(i)Rc4 output is given to Serpent as an input. Thus the ciphertext generated by Rc4 is the plaintext for Serpent and the required ciphertext is the output produced by Serpent. To decrypt the ciphertext reverse the procedures i.e. use Serpent decryption 1st and give the output to Rc4 to decrypt and generate the initial plaintext. The advantage here is that even if Serpent is broken, the plaintext cannot be recovered as there is another layer of security in the form of Rc4.
(ii)Rc4 produces a random looking ciphertext as its output which becomes a key in Serpent operations. Here, the plaintext is what is given as a plaintext in Serpent and the ciphertext is Serpent output. For decryption reverse the process. Use Serpent decryption with the Rc4 output as the key. The output generated will be the initial plaintext.
The advantage here is that even if Serpent is broken, the key remains unknown as the pseudo key (the output of Rc4) is attacked and the actual key remains safe.

5.3 Proposed Hybrid Approach

The better solution is to combine rc4 ,AES and Serpent as to get a better speed without compromising the system security.

The key issue with AES and Serpent is that it is a block cipher. Many blocks are encrypted with a single key, thus interrelating the blocks. If one block is managed to be broken, all others which share the common key can also be easily broken. Attack which is difficult against one block can be highly simplified when given multiple blocks with shared key. AES and Serpent security depends on the permutation-combination transformations that are called a number of times. On reducing this number the speed of the cipher will increase but at the same time the security will decrease and the cipher will be more vulnerable to attacks. This vulnerability needs to be compensated by other changes in the algorithm. The algorithm can be made more secure if the following is used:

->Non-linear operations:
As explained in the experimental design, nonlinear keys are much more secure compared to linear keys. They make it much more difficult to obtain the plaintext even if the keys used with different plaintexts are the same. Moreover, it is comparatively secure against a known plaintext attack.
->Make sure all the blocks in AES don't use the same key:
As explained above, AES is a block cipher and use of same key with multiple blocks is a security issue. Changing that will make the cipher highly secure.
->Make the output cipher look as random as possible (can use whitening):
Randomness ensures that there is no connection or similarity between the plaintext and ciphertext thus making it much more difficult to hack at a plaintext. Whitening is a well-known cryptographic technique which is used to mix up and combine various text or cipher characters thus giving a more random look to the original plaintext. This is a reversible process and thus can easily be imbibed in any cipher.
->Linear Transformation LT giving block Bi+1 that is the input to the next round. In the last round R7 the linear transformation is replaced with XOR operation with the last key K8 (therefore two keys are required in this round).
The concept is illustrated below:
Decreasing rounds in AES and Serpent would increase the speed. So, considering the 128 bit AES and Serpent, the number of rounds is reduced from 10 to 8 in this particular hybrid cipher. This compromises the security. To compensate, any of the above mentioned techniques could be used. This would considerably increase the security without decreasing the speed too much. This can be represented as shown below.
Fig-4 shows the part of AES,Serpent that is used in the hybrid cipher.
Unlike the 10 rounds in the original 128 bit cipher, this has only 8 rounds with the first 7 rounds having all the permutation and transformation techniques of byte substitution, row shifting, mix columns, and then finally round key addition before moving to the next round. Round 8 in the hybrid is same as the round 10 in the original AES and the round 32 in the original Serpent. It doesn't include column mixing and Linear Transformation. Similarly in the 1st round, Round key is added before the encryption rounds start.
Whitening is used to enhance the security of the cipher in the following manner. If P is the plaintext and W is the whitening value which is to be used, then a non-linear operation is used to combine P and W to generate whitened plaintext called C1. This whitening of the plaintext is done before the actual encryption. C1 looks very random as whitening plaintext P it with W will generate random looking text.Whitening can be carried out as follows. If length of whitening value W is L1 and length of the data to be encrypted P is L2 then, there can be three things that will happen.

1. L1=L2
This is the perfect case in which the length of the whitening value and the length of the data is same and the data can be whitened without a hitch. Thus C1=P+W, where '+' is any non-linear operation.
2. L1>L2
When length of the whitening value is more than the length of the data some extra amount of padding has to be hitched to the end of the actual data for whitening to go smoothly.
3. L2>L1
When the length of the plaintext data is more than the length of the whitener W, the whitener has to be used again and again with the plaintext in rotation to make sure that all the text gets whitened. Even though the whitener is used repeatedly on the same data, it would still be random and not easy to break because of the non-linear operation. Thus the size of the whitener is increased by continuously appending the whitener to itself.

The above is implemented inthe following manner:

```
Input<-P,W
If L1==L2
C1=P+W
Else if L1>L2
P1=P
While L1!=L2
P=X.P1//padding
C1=P+W
else
W1=W
While L1<L2
W=W.W1//append
if L1>L2
P1=P
While L1!=L2
P=X.P1//padding
C1=P+W
Output<-C1
```

This way, even if the cipher is broken, the plaintext cannot be recovered as the whitening added to the plaintext makes it look random. This way the attacker cannot know if the cipher has been broken as the plaintext still appears random.
Whitening operation is also pretty cheap hence it wouldn't cause a lot of overhead to the cipher. To further increase the complexity whitening value can be used with the round key for each block so that each block doesn't use the same key and the cipher becomes more secure against attack. During decryption it is a simple reverse process to remove the whitening since the operation used is non-linear invertible. This way unless the whitening values and the key used is known, the plaintext cannot be deciphered.
The new cipher is put forward as follows:

```
Input<-P,W   //generate whitening value W
C1=P+W   //whitening //C1 will act as the plaintext which is to be inputted into Rc4
Generate key S1
C2=C1(Rc4)S1
Function modifiedAES
Pass in: C2, key
State ← Input
Generate Round Key//use whitening on round key using W2
Add Round Key  For round →1 to round→5
{
SubBytes
Shift Rows
Mix Columns
AddRoundKey
}
SubBytes
Shift Rows
Add Round Key
Output ← state
Pass out: C
End Function
```

W is random. W is generated before the actual encryption process starts with Rc4. C1 (whitened plaintext) is given as the input plaintext and S1 is the key used to encrypt it. The output generated from the Rc4 is C2. This is what is fed as input to the modified AES and Serpent.

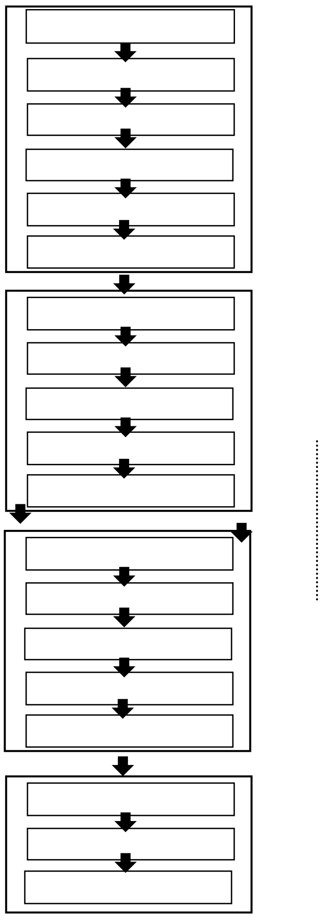


Figure 4: Encryption rounds of Hybrid Approach

Another whitener is generated called W2 to whiten the round keys. The final output from the AES / Serpent is our desired ciphertext which is C. For every round, the bits in the whitener are altered such that it can act as a different whitener for every round. Thus the round key used for every round will be different. Modifying the bits in the whitener does not incur much overhead and the resultant changes in round keys makes the cipher more secure against attacks.

Decryption is carried out by reversing the above process. For a person who has the whiteners and the necessary keys, decryption is a simple process. But without the whiteners and the key, breaking this cipher would be very tough.

6. THEORETICAL CALCULATIONS

is also relatively cheap and easy operation. A small percent of actual time can be allotted to these operations to The benchmarks used [12]:

AES=206.19=approx. 206 cycles/byte
SERPENT=742.284=approx. 742 cycles/byte
Rc4=26.97=approx. 27 cycles/byte
=>AES/Rc4=7.6 approx.And serpent/rc4= 27.59

The above statements hold for a stream of continuous data where Rc4 is much faster than AES and Serpent.

Initially there are 10 rounds. After reducing 2 rounds, the speed will increase and time will decrease by 20%. Therefore the time taken by AES now will be 20% less or 80% of the original time taken which is 165 and that is approx = 6.1(speed of Rc4). Since Rc4 is also used in the algorithm, the speed will be reduced by some factor which is determined by the speed of the Rc4.

The time taken by the new algorithm = (time of new AES algorithm) + (time taken to run Rc4) = 165+27 = 189.

The percent increase in speed of the new algorithm with respect to the original AES is calculated using this which approximately comes to 8.33%. Owing to the use of whitening operations and Linear Transformation which involve non-linear combination with the plaintext and with the round keys in AES and Serpent, there is overhead. Moreover the bit values of the whitener are changed in every round. But whitening is a cheap operation with very little overhead. Altering bit positions account for the overhead.

Thus taking the cost of the overhead into account the resultant algorithm should be at least 9% faster than the original one.

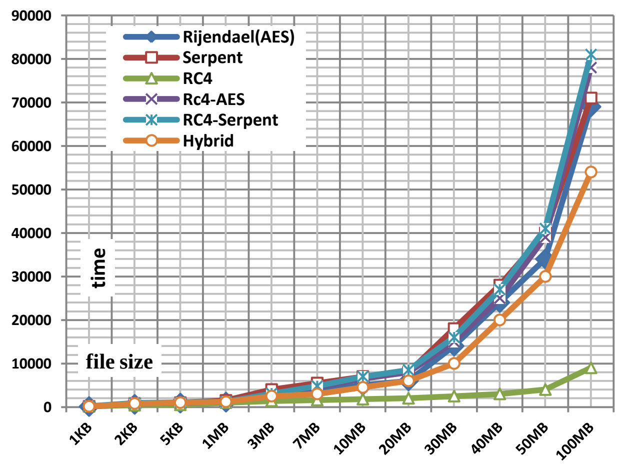
7. EXPERIMENTAL OBSERVATIONS

Table1 rates the performance of each of the individual ciphers. The performance is determined on the basis of the time taken to encrypt and decrypt the data. It is seen that RC4 is the fastest. AES_RC4 and RC4_AES take a bit more time than AES since it combines 2 ciphers with overhead. The proposed hybrid algorithm proves to be an intermediate and takes fairly less time compared to AES.

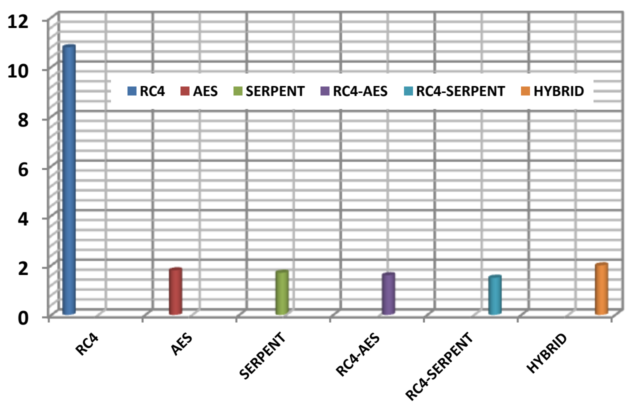
File size	Rc4	AES	AES_RC4	SERPENT-RC4	Hybrid
1KB	13	89.4	101.4	102.8	69.5
2KB	24.9	178.2	205.3	206.1	140.6
5KB	52.3	393.1	447.4	448.6	313.8
1MB	101.2	784.8	887.1	887.3	624.2
3MB	291.8	2187.8	2481.5	2480.3	1748.9
5MB	456.3	3420.8	3879.2	3880.1	2733.6
7MB	639.1	4788.5	5431.8	5430.6	3828.1
10MB	905.3	6789.75	7697.4	7697.8	5429.2
20MB	1810.3	13668.7	15481.3	15482.6	10931.6
30 MB	2715.6	20530.4	23248	23247.3	16417.7
40 MB	3615.7	27450	31069.7	31068.1	21915.6
50 MB	4520.5	34279.6	38808.1	38807.2	27413.2
100 MB	9089.4	68350.2	77420	77430.6	54678.5

Table 1: Performance of Ciphers: Time Vs File Size

The performance of the ciphers is shown in the graph below. The graph (Graph-1&2) echoes the results in the table. Minimum time is taken by Rc4 and the hybrid cipher proves to be fairly faster than AES. AES_RC4 and RC4_AES have almost the same execution time due to which the graph lines overlap.



Graph 1: Time Vs File Size



Graph 2: Ciphers Vs Throughput

The above graph compares all the ciphers based on their throughput. RC4 again tops with the maximum throughput since it encrypts maximum data in minimum time. Followed by RC4 is the hybrid algorithm which is fairly better than the other ciphers.

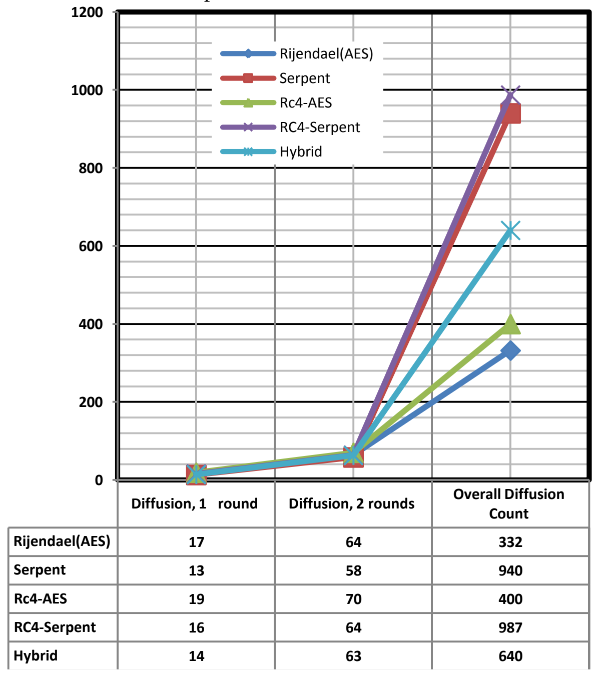
7-2 Diffusion analysis

Diffusion is made for algorithms that exhibits a strong avalanche effect taking the following cases.

- Changing one bit at a time in a plaintext, keeping key as constant.
- Changing one bit at a time in a key, keeping plaintext as constant.
- Changing many bits at a time in a key, keeping plaintext as constant.
- Changing many bits at a time in a plaintext, keeping key as constant.

As shown in Graph-3, at the end of first round ,RC4-AES again tops with the Avalanche Effect. As we observe in the Graph-3, at the end of final round and Avalanche values changes around the Serpent,RC4-Serpent and hybrid algorithm. This resulted Followed by Serpent & RC4-Serpent is the hybrid algorithm which is fairly better than the other ciphers.

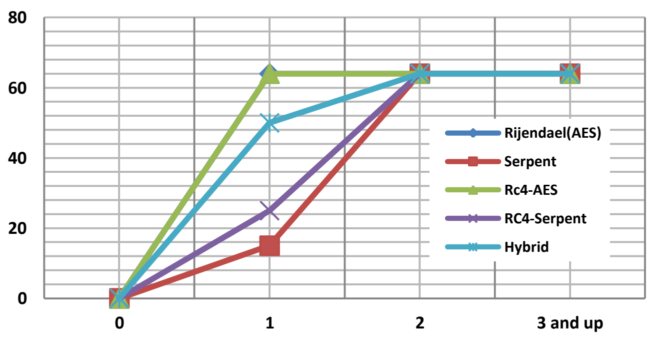
the Graphs (4-9) which illustrate the avalanche rates of all algorithms. Graph-4 compares 5 of the algorithms together to show the average amount of avalanche versus the number of rounds. The horizontal axis is for rounds one through 32. The vertical axis is the number of bits, on the average for one encryption, that change in a ciphertext, compared with a reference ciphertext, when a single bit is changed in the plaintext, compared with a reference plaintext. 16 keys were used while 128 single bits changes were made for each key, for a total of 2k encryptions for each algorithm for each round count. Graph 5-9 is a histogram of avalanche for the algorithms. On the horizontal axis are the number of bits that changed in the ciphertext after a single bit was changed in the plaintext. The vertical axis is labeled # Occur. That is the number of occurrences of encryptions when a certain number of bits were changed in the ciphertext. The total number of encryptions for this histogram was 12,800 for each round amount. The number of keys used was 100 for each round amount. The vertical scale is limited to 2000, but the peak number of occurrences was 3280 when only one bit changed in the ciphertext in the first round. The 5 Graphs (Graphs 5 - 9) in this paper were used to estimate the excess avalanche for each algorithms. The excess avalanche was estimated by dividing the total rounds for a algorithms by the number of rounds it took for the avalanche measurements to have values which are close to the optimal values.



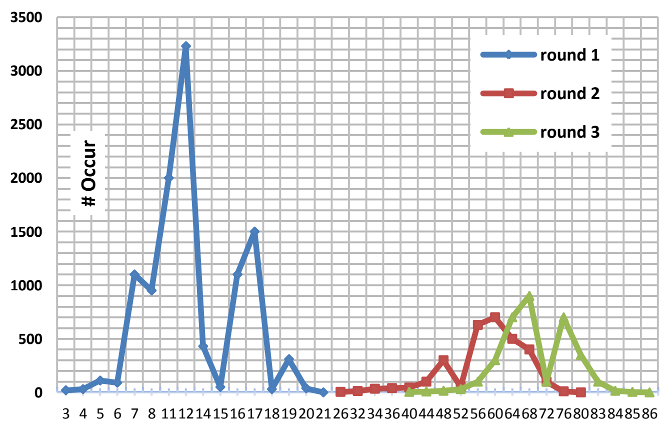
	Diffusion, 1 round	Diffusion, 2 rounds	Overall Diffusion Count
Rijendael(AES)	17	64	332
Serpent	13	58	940
Rc4-AES	19	70	400
RC4-Serpent	16	64	987
Hybrid	14	63	640

Graph 3: Results of Avalanche Effect of algorithms

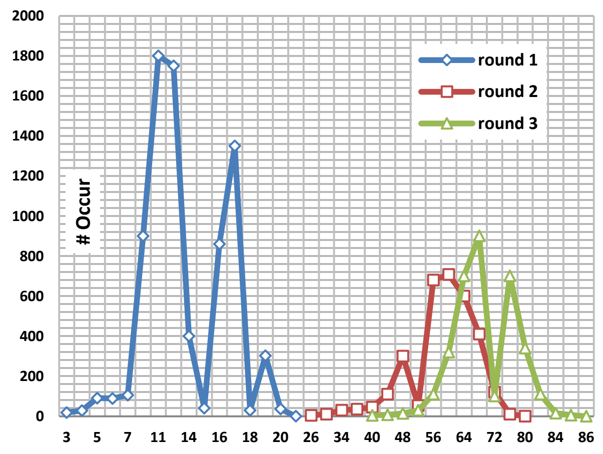
This technique has some vagueness, due to the integer quantization of rounds, and because of variable estimates of closeness to the optimum values. Excess avalanche was given the highest "weight", because this affects the security of the algorithm more than speed or code size do.



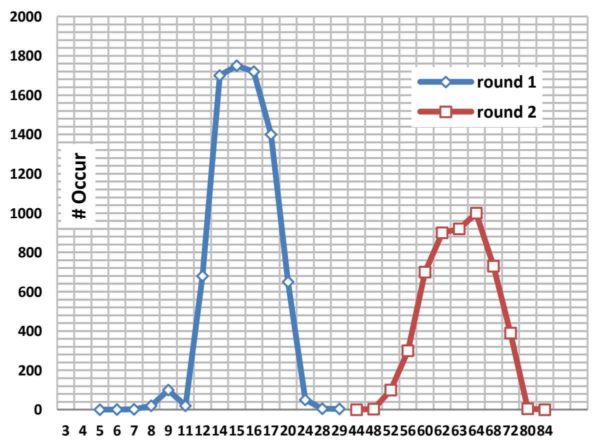
Graph 4 : Average Avalanche for the algorithms



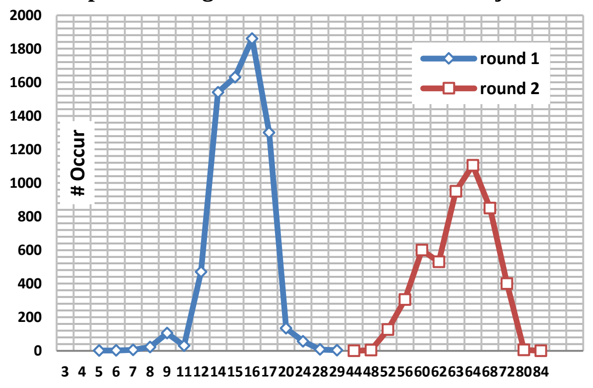
Graph 5 : histogram of avalanche for the Serpent



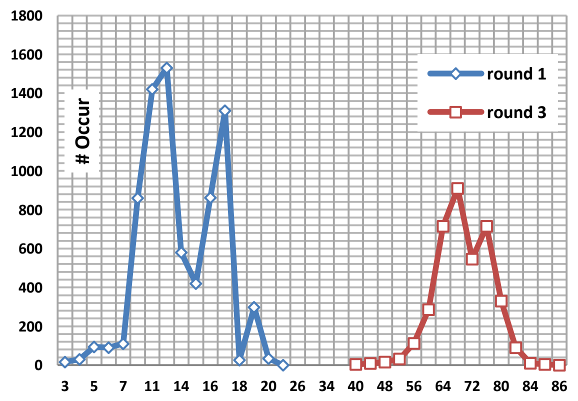
Graph 6 : histogram of avalanche for the RC4- Serpent



Graph 7 : histogram of avalanche for the Rijndael



Graph 8 : histogram of avalanche for the RC4- Rijndael



Graph 9 : histogram of avalanche for the Hybrid

8. SECURITY ANALYSIS

There are several ways to break a cipher or hack it in order to reach the significant data it encrypts. Brute force is one such technique. Brute force has been successfully applied on many ciphers to yield accurate results. Cryptanalysis is also another way through which the key can be found and used to encrypt other messages in place of the actual one. Algebraic equations can also be formulated which can generate a few bits of the key. Known plaintext attacks are also quiet common and can generate key values with high accuracy.

Most of these techniques will not work in the case of this cipher. The cipher is secure due to a number of reasons. The modified AES cipher is secured by a number of factors such as the Rc4 key and the whitening operations and Serpent Linear Transformation. The plaintext is secured many times. Security is tightened by whitening before encryption, Serpent Linear Transformation, using Rc4 encryption and by encrypting it using the modified AES operations. In order to get to the plaintext, 4 layers of security have to be broken If the modified AES is broken, then the attacker will still be left with a ciphertext which is the Rc4 input. And even if he breaks that he will still be left with the whitened plaintext. It is not enough just to get the key in order to break the cipher. Since there are so many rounds of operation involved the attacker will need to know the other agents involved like the whitener. He will need to know the keys involved in Rc4 encryption and the one used in modified AES. Breaking one key itself is a very difficult. Adding another key and a whitener tightens the security by adding more constraints.

The keys used in this case are two 128 bit keys. One is used in Rc4 and the other used in modified AES. This makes it a total of 256 bits the attacker has to break. The whitener also adds considerable number of bits to the length the attacker already has to find out in order to break the cipher.

Brute force for two 128 bit keys is out of the question. Too many combinations have to be tried out in order to generate the right key and even then the attacker will not be able to know if the key has finally been broken since he is still be left with random looking text. In order to realise if the key is broken, everything will have to be correctly evaluated including the whitener and the 2 keys. It takes a considerable amount of time to break this ensuring that brute force is not a very feasible method of breaking the cipher.

A cryptanalysis attack is slightly more plausible. But even this attack can be considered as a failure. An attacker can analyze the cipher and develop methods to break it. He can input several texts and analyze the outputs. But at every stage, the attacker will come across new and random looking plaintext which will make his analysis even more complex. The randomness involved in Rc4 and the whitener is enough to confuse the attacker and ensure the unlikely possibility of breaking the cipher by using cryptanalysis.

Several algebraic equations can be formulated to break this cipher. The number of minimum equations needed to break this cipher will be equal to the total sum of the length of the 2 keys and the whitener. Solving these equations is a very lengthy and tedious process even by using a processor. Owing to the amount of time it will take to break this cipher, and if and when the cipher is broken the attack will be ineffectual owing to the amount of time passed.

The vulnerability of the block cipher is also overcome by ensuring that the keys used in the block differ leading to the whitener whose bits are altered. Therefore if one block is broken, the other blocks remain secure from attack. A block can be broken if a known plaintext for that block is found. The attacker will need to solve for each and every block in order to break the cipher and even then it will be just one layer of security.

It is possible to study the encryption and decryption algorithm in detail in order to come up with a cipher attack that will break the cipher in a shorter time compared to brute force or cryptanalysis. All possible techniques can be combined in order to break this cipher. If all the above mentioned techniques are used together to break this cipher, breaking the cipher with accurate results is a remote possibility. This needs a lot of effort and funds in order to make this a reality. Breaking the cipher with a high cost factor is not worth the effort and thus is not feasible.

Thus the newly developed cipher is a secure cipher which can be used in various fields demanding a lot of security. The cipher will be immune to most attacks including brute force and cryptanalysis and breaking the cipher at a high cost is not feasible which heightens its security factors.

All the above mentioned reasons are sufficient to certify that the cipher is secure enough to suit the needs of most applications that need high security.

9. INFERENCE

RC4 terms up the best when it comes to speed and throughput but the rating on security scale is not very high. On security scale, AES scores better but it is definitely not as fast as Rc4. It is possible to get better and highly secure algorithms by using a combination of Rc4 and AES. The new hybrid algorithm is highly secure compared to most other ciphers and is faster than the original AES algorithm. Thus it is possible to inculcate the better parts of two algorithms into a new algorithm which fares better than the original. By reducing the number of rounds, the AES is made faster and the reduced security in this case is compensated by using the security techniques explained. The proposed hybrid cipher is seen to have a 20% improved speed compared to the original AES and a higher security compared to the original Rc4.

This highly improved cipher is easy to understand and generates secure ciphertext in a short time. It is thus a very realistic method of approach in modern applications which require security without compensating the speed. This was not possible with the individual ciphers of Rc4 and AES but the hybrid cipher is better applicable since it is a combination of both.

10. FUTURE WORK

The field of cryptography is getting more advanced due to the upcoming trends in internet network and wireless. Further work is possible in this area by combining more such ciphers. Ciphers can be combined to make them more secure, more usable and effective. This technique can be further used in all spheres wherever encryption is needed. This paper discusses how to encrypt text securely within a short time using the hybrid cipher. This can be further extended to other forms of data such as images and audio video transmissions which also require high security but without the overhead of extra time taken for encryption.

11. REFERENCES

[1] William Stallings “Cryptography and Network Security”,3rdEdition, Prentice-Hall Inc., 2005.

[2] Janakiraman V S, Ganesan R, Gobi M “HybridCryptographic Algorithm for Robust Network Security”ICGST- CNIR, Volume (7), Issue (I), July 2007

[3] HimaniAgrawal, Monisha Sharma, “Implementation and analysis of various symmetric cryptosystems”, Indian J.Sci, Technol. Vol.3, Issue 12, pp:1173-1176, domain sire : http://www.indijst.org.

[4] S. Harris, "Exploring Cipherspace: Combining stream ciphers and block ciphers", presented at IACR Cryptology ePrint Archive, 2008, pp.473-473. (http://eprint.iacr.org/2008/473.pdf)

[5] Rick Wash, Lecture Notes on Stream Ciphers and Rc4 (http://www.rickwash.com/papers/stream.pdf)

[6] AllamMousa and Ahmad Hamad (2006), "Evaluation of the RC4 Algorithm for Data Encryption", International Journal of Computer Science & Applications Vol. 3, No.2 , June 2006, pp 44-56.(http://www.tmrfindia.org/ijcsa/V3I24.pdf)

[7] HussamKassem, Hamad Saber, “Better Performances of RC4 Ciphering Using New Algorithm”, Australian Journal of Basic and Applied Sciences, 5(4): 127-134, 2011 ISSN 1991-8178 (http://www.insipub.com/ajbas/2011/127-134.pdf)

[8] Serge Mister, Stafford E. Tavares “Cryptanalysis of Rc4 like Ciphers”, Proceedings, Workshop in Selected Areas of Cryptography, SAC '98. 1998.

[9] James Nechvatal, Elaine Barker andLawrence Bassham, “Report on theDevelopment of the AdvancedEncryption Standard (AES)”, Computerand Security Division, National Instituteof Standards and Technology (NIST), USDept. of Commerce.

[10] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AESAlgorithm Submission, September 3, 1999.

[11] J. Daemen and V. Rijmen, The block cipherRijndael,Smart Card research and Applications, LNCS 1820,Springer-Verlag, pp. 288-296.

[12] Scott R. Fluhrer , ItsikMantin , Adi Shamir, Weaknesses in the Key Scheduling Algorithm of RC4, Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography, p.1-24, August 16-17, 2001 (http://merlot.usc.edu/cs531-s12/papers/Fluhrer01a.pdf)

[13] R. Anderson, E. Biham, L. Knudsen, “The Case for Serpent”, TheThird Advanced Encryption Standard Candidate Conference, April 13–14, 2000, New York, USA (proceedings available from http: //csrc.nist.gov/encryption/aes/round2/conf3/aes3conf.htm), 2000.

[14] R. Anderson, E. Biham, L. Knudsen, “Serpent and Smartcards”, SmartCard Research and Applications, Proc. 3rd InternationalConferenceCARDIS '98, Louvain-la-Neuve, Belgium, September 14–16, 1998.Lecture Notes in Computer Science, Volume 1820, Springer, 2000.

[15] R. Anderson, E. Biham, L. Knudsen, “Serpent: A Proposal for theAdvanced Encryption Standard”, The First Advanced Encryption Standard (AES) Candidate Conference, Ventura, California, August 20–22, 1998 (http://www.cl.cam.ac.uk/~rja14/serpent.html), 1998..

[16] J. Sugier,Implementing Serpent Cipher in Field Programmable Gate Arrays, PolandICIT 2011 The 5th International Conference on Information Technology

[17] Jovan Dj. Goli´c, Linear statistical weakness of alleged RC4 keystreamgenerator, Advances in Cryptology—

EUROCRYPT ’97 (Konstanz),Lecture Notes in Comput. Sci., vol. 1233, Springer, Berlin, 1997,pp. 226–238. MR MR1603060

[18] ItsikMantin, Analysis of the stream cipher RC4, Master’s thesis, TheWeizmann Institute of Science, 2001.

[19] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of AppliedCryptography, CRC press, 2001.

[20] John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, “Improved Cryptanalysis of Rijndael, Fast Software Encryption”, 2000 pp213–230

[21] RSA Laboratories, What is Rc4(http://www.rsa.com/rsalabs/ node.asp?id=2250)

[22] “Survey and Benchmark of Stream ciphers for Wireless Sensor Networks” N. Fournel, M. Minier, S. Ubeda - LIP, ENS Lyon ,France -May 10, 2007 (http://wistp2007.wistp.org/fileadmin/wistp /wistp 2007/Slides2007/Day2/WISTP2007-SmallDevices-p3.pdf)