Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher

Shaaban Sahmoud, Wisam Elmasry and Shadi Abudalfa
Department of Computer Engineering, Islamic University of Gaza

Abstract: *In this paper we developed a more powerful algorithm for cryptography. This algorithm is based on AES to generate different sub keys from the original key and using each sub key to encrypt one AES block. We used AES to protect our design from structural analysis, because AES is very resistance to this type of attacks. From other side generating many keys from symmetric key resists modern attacks on symmetric-key cryptography. Experimental results are shown in this paper to demonstrate the effectiveness of our design.*

Keywords: *Symmetric Key Cryptosystem, AES, Keys generation, Cryptanalysis attacks.*

Received April 9, 2011; Accepted March 18, 2012

1. Introduction

Cryptography is the science and art of creating secret codes, and cryptanalysis is the science and art of breaking those codes. Although in the past cryptography referred only to the encryption and decryption of messages using secret keys, today it is defined as involving three distinct mechanisms: symmetric-key encipherment, asymmetric-key encipherment, and hashing.

Symmetric-key encipherment [8] uses a single secret key for both encryption and decryption. In asymmetric key encipherment [9] there are two keys instead of one: public key used for encryption and private key used for decryption. In hashing [10], a fixed-length message digest is created out of a variable-length message.

Symmetric ciphers are divided into two broad categories: stream ciphers and block ciphers.

In a stream cipher, encryption and decryption are done one symbol (such as a character or a bit) at a time. We have a plaintext stream, a ciphertext stream, and a key stream. In block cipher, a group of plaintext symbols of size m (m>1) are encrypted together creating a group of ciphertext of the same size. Based on the definition, in a block cipher, a single key is used to encrypt the whole block even if the key is made of multiple values.

In this paper, our study concentrates on symmetric-key block ciphers, and we use the Advanced Encryption Standard (AES) as the most significant standard for block ciphers because AES provides strong encryption and has been selected by NIST as a Federal Information Processing Standard in 2001, and in 2003 the U.S. Government announced that AES is secure enough to protect classified information up to the top secret level, which is the highest security level and defined as information which would cause "exceptionally grave damage" to national security if disclosed to the public.

1.1. Advanced Encryption Standard [1]

The AES is a symmetric block cipher that encrypts and decrypts a data block of 128 bits, which can be represented by 4×4 matrix of bytes.

AES uses 10, 12, or 14 rounds N_r. The key size is variable, which can be 128, 192 or 256 bits, depends on the number of rounds. Figure 1 shows the general design for the encryption algorithm; the decryption algorithm is similar, but the rounds are applied in reverse order. The key expansion derives N_r+1 round keys k_0 to k_{N_r} from the master key k.

The round function, repeated N_r-1 times, is composed of four basic transformations, all linear except the first one:

- *SubBytes*: each byte in the 4×4 matrix is updated using an 8-bit substitution box, the S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over $GF(2^8)$, known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points.
- *ShiftRows*: a linear mapping that rotates on the left all the rows of the current matrix (0 for the first row, 1 for the second, 2 for the third and 3 for the fourth).
- *MixColumn*: another linear mapping represented by a 4×4 matrix chosen for its good properties of diffusion. Each column of the input matrix is

multiplied by the MixColumns matrix in GF(2^8) that provides the corresponding column of the output matrix.

- *AddRoundKey*: the round key is combined with the state (4×4 matrix). For each round, a round key is derived from the main key using key schedule; each round key is the same size as the state. The round key is added by combining each byte of the state with the corresponding byte of the round key using bitwise XOR.

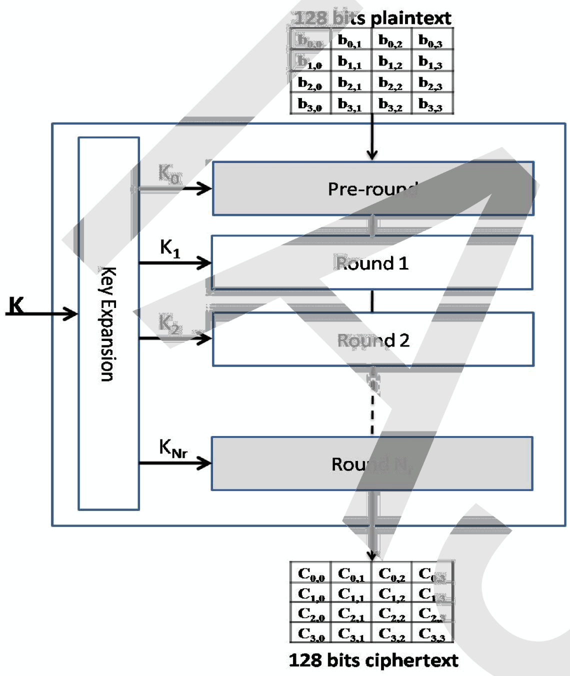
128 bits plaintext
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$
K
Key Expansion
K_0
K_1
K_2
K_{Nr}
Pre-round
Round 1
Round 2
Round N_r
$C_{0,0}$ $C_{0,1}$ $C_{0,2}$ $C_{0,3}$
$C_{1,0}$ $C_{1,1}$ $C_{1,2}$ $C_{1,3}$
$C_{2,0}$ $C_{2,1}$ $C_{2,2}$ $C_{2,3}$
$C_{3,0}$ $C_{3,1}$ $C_{3,2}$ $C_{3,3}$
128 bits ciphertext

Figure 1. General design of AES encryption cipher.

Each transformation takes a state and creates another state to be used for the next transformation or the next round. The pre-round section uses only one transformation (*AddRoundKey*); the last round N_r uses only three transformations (*MixColumn transformation is missing*).

1.2. AES Performance

AES is the best and strongest cryptology algorithm, because of three areas: security, cost, and implementation. AES proved that it secure against many attacks such that brute-force, differential and linear attacks.

The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. So using the Brute-Force attack is almost impossible.

The strong diffusion and confusion provided by the combination of the SubBytes, ShiftRows, and MixColumns transformation removes any frequency pattern in the plaintext. Numerous tests have failed to do analysis of the ciphertext.

From other side, AES can be easily implemented using cheap processors and a minimum amount of memory.

Although of these facts there are many papers recently try to attack AES using different cryptanalysis techniques. We will present these attacks briefly in next sub section.

1.3. Analysis on AES

There are many cryptanalysis systems are designed to attack AES. We divided these attacks on AES into two categories: attacks need the same key, and attacks don't need the same key.

1.3.1. Attacks Need the Same Key

In April 2005, D.J. Bernstein announced a cache-timing attack that he used to break a custom server that used OpenSSL's AES encryption [2].The custom server was designed to give out as much timing information as possible (the server reports back the number of machine cycles taken by the encryption operation), and the attack required over 200 million chosen plaintexts. Some say the attack is not practical over the internet with a distance of one or more hops; Bruce Schneier called the research a "nice timing attack" [20].

In October 2005, Dag Arne Osvik, Adi Shamir and Eran Tromer presented a paper demonstrating several cache-timing attacks against AES [18]. One attack was able to obtain an entire AES key after only 800 operations triggering encryptions, in a total of 65 milliseconds. This attack requires the attacker to be able to run programs on the same system that is performing AES.

Tadayoshi Kohno wrote a paper entitled "Attacking and Repairing the WinZip Encryption Scheme" [16] showing possible attacks against the WinZip implementation (the zip archive's metadata isn't encrypted).

Warren D. Smith describes a new powerful form of linear cryptanalysis [21]. It appears to break AES. A small algorithm can quickly determining AES-256 keys from plaintext-ciphertext pairs exists. The attack's runtime is faster than exhaustive key search.
Joseph Bonneau and Ilya Mironov [5] describes several novel timing attacks against the common software implementation of the AES cipher with the same key.

1.3.2. Attacks Don't Need the Same Key

Many attacks called side-channel attacks do not attack the underlying cipher and so have nothing to do with its security, but attack implementations of the cipher on systems. There are several such known attacks on certain implementations of AES. The most common way to attack block ciphers is to try various attacks on versions of the cipher with a reduced number of rounds. AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. 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 [7].

A family of Impossible Differential Attacks [19, 11, 23, 3, 4, 12, 13] was proposed in different papers but all of them can't exceed attack on 7-Round 128 bit AES and 8-Round 192 or 256 bits AES. Huseyin Demirci and Ali Aydın Selcuk [6] developed Meet-in-the-Middle Attack on 8-Round AES so it can't attack full-round AES.

Some other papers discussed weakness of Key Schedules [17] which may be used in attacking the block cipher system, but no attacks, including key-schedule attacks, on the full-round AES algorithm have been found and published until now.

1.4. Related Work

There are few papers try to enhance the security of AES because most of the AES attacks appeared recently. In 2011 A. Sekar, S. Radhika and K. Anand [22] propose a method to enhanced the strength of the AES algorithm by increasing the key length to 512 bit and thereby the number of rounds is increased in order to provide a stronger encryption method for secure communication. Code optimization is done in order to improve the speed of encryption and decryption using the 512 bit AES. This method don't modify the structure of AES but only increase the number of rounds and so the attacks which need the same key are still serious to this algorithm at the same time this algorithm increase the processing time which will limit the use of AES in real application.

Mark Karpovsky, Konrad J. Kulikowski, and Alexander Taubin [15] developed two architectures for protecting hardware implementation of AES against side-channel attacks known as Differential Fault Analysis attacks. The first architecture, which is efficient for faults of higher multiplicity, partitions the design into linear (XOR gates only) and nonlinear blocks and uses different protection schemes for these blocks. They protected the linear blocks with linear codes and the nonlinear with a complimentary nonlinear operation resulting in robust protection. The second architecture uses systematic nonlinear (cubic) robust error detecting codes and provides for high fault detection for faults of low and high multiplicities but has higher hardware overhead.

Aida Janadi and D. Anas Tarah tried some methods to increase the immunity of block ciphers in general (and AES in special) against the algebraic attacks which become more dangerous after XSL algorithm developing [14]. These algorithms enhance the security of AES algorithm against side-channel attacks but the other attacks still be possible.

1.5. Contributions

We can conclude that, there are a lot of attacks on AES using different cryptanalysis techniques. so to resist these attacks we concerned on protection the symmetric key used by AES, because we believe that there is a good protection of AES structure against attacks and the most important attacks use the same cipher key for all encrypted blocks to complete the cryptanalysis steps, because it attack the full round AES. For example, Daniel J. Bernstein need 2×10^8 plaintext-ciphertext-time triples on the same key. This performs to attack failed if we changed the key with each block encryption. Depending on this idea we designed our algorithm.

We developed a cryptosystem that changing the key for every encrypted block by generating sub keys from the original key using the same design of AES. Next section describes design of our cryptosystem. In section 3 we show the analysis of our cryptosystem. Finally section 4 presents the performance evaluation of our design. Section 5 concludes the paper.

2. Variable Key Cipher Design

2.1. Approach

The Proposed model to enforce AES combines the two famous models for encryption the block cipher and stream cipher effectively so as to increase the security of the cipher. We propose here a new simple but more powerful algorithm for cryptography. We will name it as Variable Key Cipher. It based on AES algorithm but with some differences. It is a symmetric block cipher like AES that encrypts and decrypts a data block of 128 bits , but unlike the AES which use the same secret key to encrypt and decrypt many blocks. The proposed algorithm generate different sub keys from the original key then each sub key is used to encrypt one AES block cipher to make cryptanalysis with the attacks described in sub section 1.3 failed. It looks like combining the ideas of both block cipher and stream cipher. It consists of two stages the first is sub key generation and the second is encryption/decryption process. Figure 2 demonstrates the main idea of this algorithm.

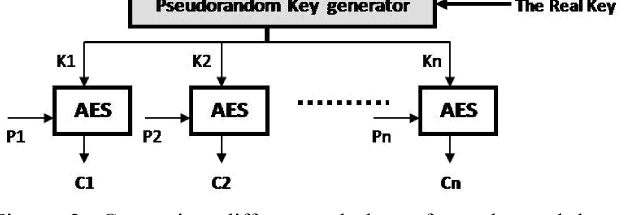

Figure 2. Generating different sub keys from the real key in proposed algorithm.

In the encryption process we use the AES cipher where in the decryption process the same steps of encryption will be used but the inverse AES cipher will be used. The main steps of the proposed algorithm are explained in these steps:

1. Input the Symmetric key (we name it in the paper as Real key).
2. Generate from the real key the first Pseudorandom sub key K_1 using Pseudorandom Key generator (we will explain it in details in the next subsections).
3. Encrypt the first block of plaintext P_1 by the AES using K_1 which generated in the previous step.
4. Generate from the real key the second Pseudorandom sub key K_2 using Pseudorandom Key generator.
5. Encrypt the second block of plaintext P_2 by the AES using K_2.
6. Generate the next sub key to encrypt the corresponding block of plaintext and repeat this step until encrypt the last block in plaintext P_N with sub key K_N.

The proposed algorithm generate stream of sub keys to encrypt stream of plaintext blocks each block encrypt with different sub key but all sub keys can be obtained by the shared real key. The decryption process is the reverse cipher of the encryption process with the same real key that will generate the same sub keys stream each one to decrypt the corresponding ciphertext block, it means that when specified sub key K_i used to encrypt the plaintext block P_i to generate C_i then the same key K_i will used to decrypt the generated block cipher C_i to generate P_i. We can consider the real key as shared symmetric key and it can be shared between encryption site and decryption site through one of symmetric key distribution methods. The real key size in our proposed algorithm is 128 bits. Figure 3 shows the Encryption/Decryption process in the proposed algorithm.

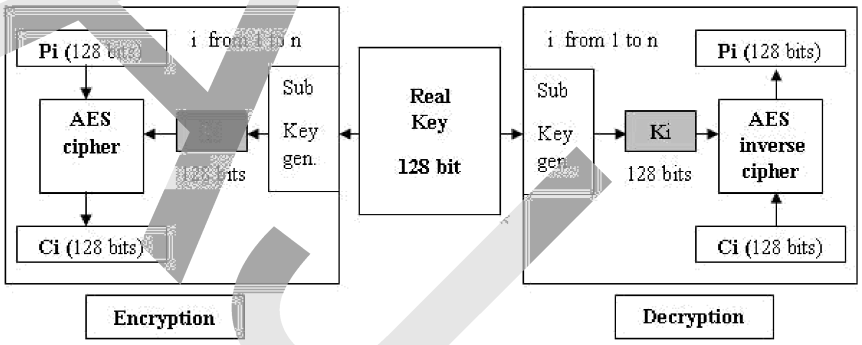

Figure 3. Encryption and decryption process in the proposed algorithm.

The most important step in the algorithm is pseudorandom sub keys generation from the real key. It must be very secure to make the gain of the real key from the generated stream of sub keys impossible, and also must K_1 has no relation with K_2 and all other sub keys to prevent the attacker from expect other sub keys from one sub key, we propose two techniques that based on AES to make this in the next sub sections. The two techniques are simple but more powerful to generate pseudorandom sub keys from the real key. The first technique based on the AES with feedback process to generate the stream of sub keys, the second technique is also based on the AES but use counter.

2.2. First Pseudorandom Sub Keys Generator (AES with Feedback)

The first technique based on the AES with feedback from the previous cipher block to generate the stream of sub keys from the real key, these steps explain the algorithm:

1. Use a copy of the real key and refer to it as IV.
2. Use IV as plaintext and Encrypt it by the AES using the real key and let the result name is C_1.
3. Use C_1 as the first sub key (K_1).
4. Xor C_1 with IV and use it as plaintext and Encrypt it by the AES using the real key let the result name is C_2.
5. Use C_2 as the second sub key (K_2).
6. Xor the previous cipher block with IV and use it as plaintext and Encrypt it by the AES using the real key to obtain the next sub key, then repeat this step until generate the last sub key K_N.

This technique generate the sub keys as following: the first sub key has the relation: $k_1=E_k(IV)$ and the rest of sub keys have the relation: $K_i= E_k(C_{i-1} \text{ xor } IV)$ where the E_k means the Encryption process of AES using the real key. So the real key generates all the sub keys that will be used later to encryption plaintext blocks or decryption ciphertext blocks. Figure 4 shows the design of this technique.

2.3. Second Pseudorandom Sub Keys Generator (AES with Counter)

The second technique is also based on the AES but use a counter to change the plaintext for each process of sub key generation instead of feedback process, these steps explain this technique:

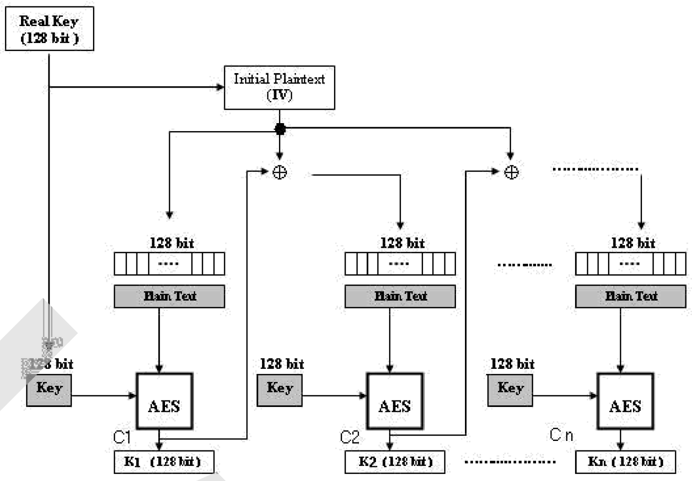

Figure 4. AES with feedback pseudorandom key generator.

1. Use a copy of the real key and refer to it as IV.
2. Use IV as plaintext and Encrypt it by the AES using the real key let the result name is C_1.
3. Use C_1 as the first sub key (K_1).
4. Add fixed value (it will be discussed in details in next sub section) to the IV and use the result IV_1 as plaintext and Encrypt it by the AES using the real key let the result name is C_2.
5. Use C_2 as the second sub key (K_2).
6. Add fixed value to the previous initial plaintext and use the result as plaintext and Encrypt it by the AES using the real key to obtain the next sub key, then repeat this step until generate the last sub key K_N.

This technique generate the sub keys as following: the first sub key has the relation: $k_1=E_k(IV)$ and the rest of sub keys have the relation: $K_i=E_k(IV_{i-1} + \text{value})$ where the E_k means the Encryption process of AES using the real key. So the real key generates all the sub keys that will be used later to encryption plaintext blocks or decryption ciphertext blocks. Figure 5 shows the design of this technique.

2.4. Security of Sub Keys Generation Techniques

The most important goal of cryptography is the confidentiality of data. From the two proposed techniques to generate sub keys from the real key we see that AES is used because AES cipher is the most strong and high diffusion and confusion cipher and so generate approximately random values.

In the first technique we Xor the initial plaintext IV to the previous sub key K_i to change as much as possible bits in the new plaintext which will make more differences in the resulting next sub key. Note

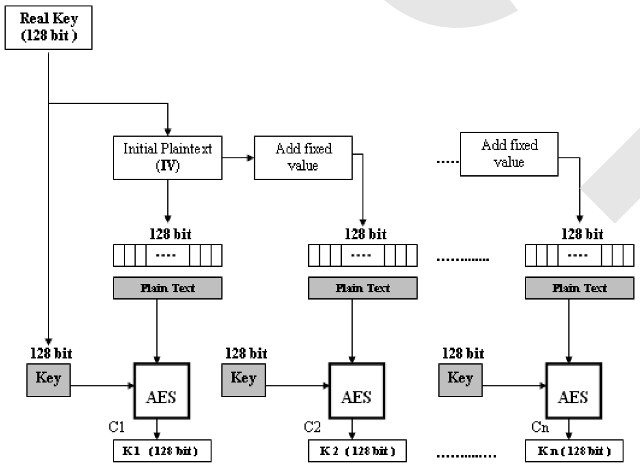

Figure 5. AES with Counter Pseudorandom Key Generator.

that unlike the CFB in modes of operation we didn't input the previous sub key directly to the AES cipher because if we did the attacker if success in getting certain sub key it may get some information about the next sub key, so we Xor the previous sub key with IV to prevent the attacker from getting any information about the next key. This will make this algorithm more secure against any try to know or analyze sub keys. In the second technique we will select a fixed value for all sub keys generation that has this main property: when added to IV gives changing as much as possible bits in the new plaintext which will make more differences in the resulting next sub key. We suggest this fixed value to be number three to be added to each previous IV, because when number three is added it always change at least two bits as shown in Figure 6. We know that when change only one bit in the plaintext of AES will affect many bits in the ciphertext as shown in Figure 7. So when add number three it will make more changes and randomize on resulting sub keys. Finally note that we always use the real key as the key for AES ciphers in all sub keys generation.

3. Variable Key Cipher Analysis

3.1. Variable Key Cipher Versus AES

Our algorithm is based on AES but not updated version of AES. Our algorithm is new simple algorithm that uses AES twice. The first when we generate the sub keys stream from the real key and the second when we encrypt plaintext blocks or decrypt ciphertext blocks. This is another advantage to our algorithm because using AES twice will increase the immunity against modern attacks. The another difference between our algorithm and AES is that our algorithm use different sub keys each for the corresponding block but in AES the key is the same for all blocks.

Number	Add 3	Number of changed bits
0000	0011	2
0001	0100	2
0010	0101	3
0011	0110	2
0100	0111	2

Figure 6. Effect of add 3 to each number.

```
Plaintext 1:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plaintext 2:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01
Ciphertext 1: 63 2C D4 5E 5D 56 ED B5 62 04 01 A0 AA 9C 2D 8D
Ciphertext 2: 26 E3 9B BC A1 9C 0F B7 C7 2E 7E 30 63 92 73 13
```

Figure 7. one bit in the plaintext of AES will affect many bits in the ciphertext.

3.2. Variable Key Cipher is Hybrid Algorithm

Our algorithm is symmetric key block cipher that combines the benefits of both block cipher and stream cipher. The benefits of block cipher that it is using to both small and large messages and has only one shared parameter which is the shared symmetric key. The benefit of stream cipher that it encrypts or decrypts the messages with different keys which makes attacks more difficult. Our algorithm has all these benefits because it still symmetric block cipher with one shared parameter which is the real key but we generate from it different sub keys to make modern attacks very difficult.

3.3. Modes of Operation Versus Proposed Sub Keys Generation Techniques

Modes of operation have been devised to encipher text of any size using either DES or AES. There are five modes ECB, CBC, CFB, OFB, and CTR .Some of these modes seems like our proposed sub keys generation techniques for example CFB seems like first proposed technique. Where CTR seems like second proposed technique. We will explain the difference between them here. The Modes of operation don't add any effective type of security to the algorithm it used, because it's only tool for handling variable size plaintext. For example in CFB mode, if we know some pairs of plaintext-ciphertext we can easily get the block enter the AES and the resulting block which will make the problem like ordinary AES. But our proposed techniques add a new layer of security to the AES to make it more resistance to modern attacks.

3.4. Initialization of Proposed Algorithm

In proposed algorithm we use two AES cipher for each plaintext block; the first AES is used to generate the sub keys where the second AES is used to encrypt the plaintext block with the sub key generated from the first AES. As result the time of encryption in proposed algorithm will be twice the time of the AES. Really we can ignore this increasing in time because the huge progress in processors speed and parallel computers which will make the execution of cryptography algorithm very fast; also if we compare our proposed algorithm with other algorithms we can see that our proposed algorithm uses low number of ciphers. For example PGP pseudorandom generator use seven encryption ciphers and ANSI X9.17 PRNG uses three 3DES ciphers where our proposed algorithm uses only two AES ciphers. Although that we introduce here a simple initialization method to decrease the time of proposed algorithm to be approximately like AES. The following steps explain this method which works approximately like buffer that generate and arrange the keys to be used directly without waiting:

1. Generate from the real key N (128-bit) sub keys using one of the two proposed sub keys generation techniques.(N selected to be always greater than the message size in bits divide by 128 bits, so any message can be encrypted directly without losing

time in generating sub keys or round keys of the generated sub keys)
2. Generate from each sub key its 10 (128-bit) round keys.
3. Store each sub key followed by its 10 (128-bit) round keys in queue with N*11 elements (the sub key it self and the 10 round keys generated from it which will be 11 elements must be stored for N generated sub keys, so the size of the queue will be 11*N).
4. For each Encryption process:
   a. Divide the message which will be encrypted into M 128-bit parts (so N must be greater than M).
   b. Encrypt the M parts of the message with first M*11 elements from the queue.
   c. If the message sends successfully:
      - Delete the used elements from the queue.
      - Generate from the real key new M (128-bit) sub keys using one of the two proposed sub keys generation techniques and generate from each sub key its 10 (128-bit) round keys (To compensate the used and deleted elements in the queue), so we have new M*11 elements.
      - Add the new M*11 elements to the queue.

Note that the last sub step in step 4 is used to be ready for next encryption message.

As we can see if using this method in the proposed algorithm the time of the execution will be very close to the execution time of AES. Also this initialization method will help in re-encrypting blocks if there is losing in number of blocks.

4. Proposed Algorithm Performance and Results

We will analyze the performance of our algorithm in three directions: Security, Time and Complexity. In each of them we will compare our algorithm with AES to prove the strength of our algorithm.

4.1. Security of Proposed Algorithm

The AES is very confidential because it has high differences in generated sub keys. Making attacks against the proposed algorithm impossible or very difficult, because we know that attacks in the first category (see sub section 1.3.1) depends on high number of plaintext-ciphertext block pairs and must these pairs encrypted with the same key. Our proposed algorithm generates different sub keys for each plaintext block so making this type of attack will be impossible. Also attacks of the second category (see sub section 1.3.2) will be extremely difficult, because if the attacker success in getting one of the sub keys, he will not be able to use it to decrypt the next block because it encrypted with another sub key and also making prediction of the next sub key is impossible, because he don't know the real key or even the initial plaintext (IV). Furthermore, we make another security layer on the real key which will make it more hidden and secure, in other words, even if the attacker success to launch an exhaustive search attack to break AES it will gain only a sub key and will never get the real key. So our proposed algorithm is secure against the modern attacks described in sub section 1.3. Also because our algorithm based on AES and uses it twice, we can say that our algorithm is also secure against brute force attack, differential attack and linear attack.

4.2. Time and Complexity

Here we will present the elapsed time which is the amount of time needed for our algorithm to generate all sub keys and then encrypt/decrypt corresponding blocks. The results reported are extracted on a machine with 512 MB of main memory and a 2.6 GHz Pentium 4 processor running Windows XP sp2. We do our experiments using Java Cryptography Extension (JCE) in J2SDK 1.4.2. We implement four classes: the first is to implement the AES, the second to implement the first technique to generate sub keys, the third is to implement the second technique to generate sub keys and the fourth is to implement the encryption/decryption process using the generated sub keys. We will represent here the variable key cipher using the first technique to generate sub keys as VKC1 and the variable key cipher using the second technique to generate sub keys as VKC2. In each experiment we encrypt/decrypt a text file using AES, VKC1 and VKC2 separately and find the elapsed time in each case with millisecond timer. Table 1 shows the resulting elapsed time for different sizes of text file in Kbytes when using AES, VKC1 and VKC2 respectively.

We can note from table 1 that the results of the encryption process is the same as the results of the decryption process and also the results of VKC1 is the same as the results of VKC2 that because encryption and decryption process have the same number of steps and also in other side VKC1 and VKC2 have the same number of steps. From the table 1 we can see that the elapsed time for the same file size differs from AES to VKC (VKC1 or VKC2). The elapsed time for file sizes in range (1 to 64 Kbytes) as following: the VKC record twice the elapsed time for AES and that make sense because as we say before that VKC has two stages instead one in AES, the first to generate sub keys and the second for encryption or decryption blocks, the second stage take same time with elapsed time for AES. So the elapsed time for VKC is equal to sum of time needed for generates the sub keys stream and the elapsed time for AES for that file size. The elapsed time for file sizes in range (128 to 1024 Kbytes) as

Table 1. Experiments Results.

File size in Kbytes	**Encryption Process**			**Decryption Process**		
	Elapsed Time (millisecond)			**Elapsed Time (millisecond)**		
	AES	VKC1	VKC2	AES	VKC1	VKC2
1	16	32	32	16	32	32
2	18	46	46	18	46	46
4	20	47	47	20	47	47
8	23	62	62	23	62	62
16	47	78	78	47	78	78
32	60	125	125	60	125	125
64	68	140	140	68	140	140
128	78	219	219	78	219	219
256	94	328	328	94	328	328
512	190	580	580	190	580	580
1024	315	780	780	315	780	780

following: the VKC record more than twice elapsed time for AES and that due to high number of sub keys we want to generate in this case. But we can say that even with this increasing amount of time the elapsed time for VKC is still in milliseconds and in 1Mbytes file size (more than 64000 sub keys) the elapsed time for VKC is 780 ms (< 1 second). Finally the proposed initialization method in sub section 3.4 will help us to decrease the elapsed time significantly.

Figure 8 shows the plot of elapsed time for AES and VKC versus different file sizes. Note that red '+' represent AES and green '×' represent VKC (VKC1 OR VKC2).

Our algorithm is more complex than AES because we have here in our algorithm two stages instead of one in AES. But we can ignore this increasing in complexity for two reasons: the first due to the huge progress in processors speed and parallel computers which will make the execution of cryptography algorithm very fast; the second if we compare our proposed algorithm with other algorithms that used key generation, we can see that our proposed algorithm has very low complexity. For example PGP pseudorandom generator use seven encryption ciphers and ANSI X9.17 PRNG uses three 3DES ciphers where our proposed algorithm uses only two AES ciphers.

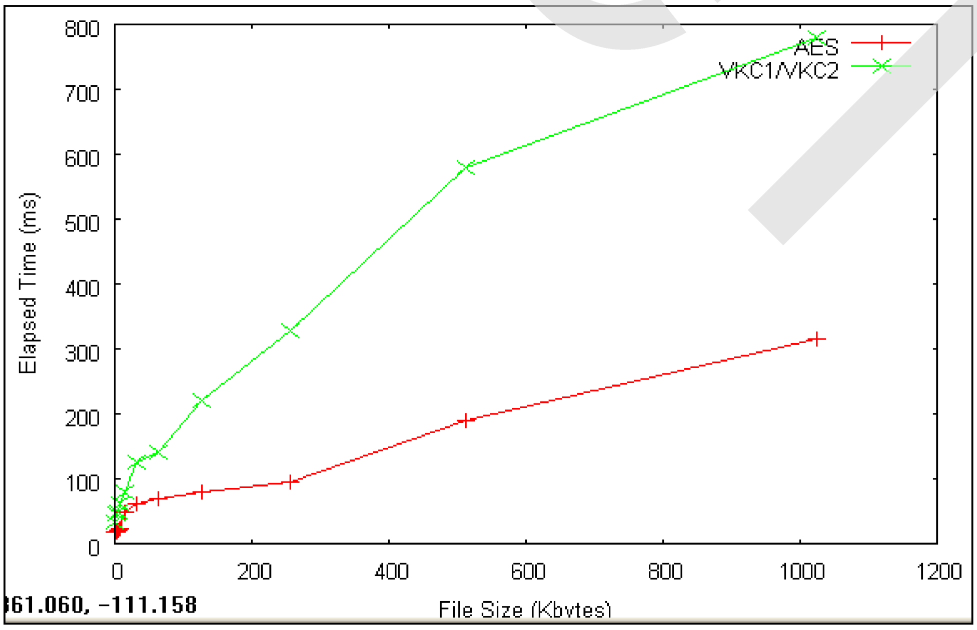

Figure 8. Elapsed time in millisecond versus file sizes in Kbytes.

5. Conclusion

We described and analysed the symmetric-key encipherment by AES in detail, and provided an overview of AES analysis papers. In this paper we presented a powerful algorithm for cryptography. This algorithm is based on AES as a modern symmetric-key block cipher to generate different sub keys from the symmetric real key and using each sub key to encrypt one AES block. We name it Variable Key Cipher (VKC) and present the general idea of our algorithm. We also present in details two techniques for sub keys generation, prove the security of these generators to prevent prediction or getting another sub key from available one and also present initialization method to make sub keys generation faster. The experiments results shows that our algorithm is slower and has more complexity to AES, but we can really ignore this for two reasons: first that the increasing in time and complexity in our algorithm can be neglected specially when we using nowadays computers and the second to get the benefit that our algorithm make the modern attacks on symmetric key cipher extremely difficult.

References

[1] Advanced Encryption Standard, National Institute of Standards and Technology (US), URL:http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[2] Bernstein, D., "Cache- timing attack on AES", Document ID: cd9faae9bd5308c440df50fc26a517b4, URL: http://cr.yp.to/papers.html#cachetiming, April 2005.

[3] Biham, E., Dunkelman, O., and Keller, N., "Related-Key Boomerang and Rectangle Attacks" Springer-Verlag, Advances in Cryptology, *proceedings of EUROCRYPT 2005*, Lecture Notes in Computer Science 3557, pp. 507–525, 2005.

[4] Biryukov, A., "The Boomerang Attack on 5 and 6-round AES", Springer-Verlag, *proceedings of Advanced Encryption Standard 4*, Lecture Notes in Computer Science 3373, pp. 11–16, 2005.

[5] Bonneau, J., and Mironov, I., "Cache-Collision Timing Attacks Against AES", Cryptographic Hardware and Embedded Systems—CHES 2006, pp. 201–215, 2006.

[6] Demirci, H., and Selcuk, A., "Meet-in-the-Middle Attack on 8-Round AES", International Association for Cryptologic Research, FSE 2008, LNCS 5086, pp. 116–126, 2008.

[7] Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., and Whiting, D., "Improved cryptanalysis of Rijndael", *Springer-Verlag Berlin Heidelberg*, vol. 1978, pp. 213–230, 2001.

[8] Forouzan, B., "Traditional Symmetric-Key Ciphers", *in Introduction to Cryptography and Network Security*, 1st ed. New York: McGraw-Hill, 2008, pp.55-96.

[9] Forouzan, B., "Asymmetric-Key Cryptography", *in Introduction to Cryptography and Network Security*, 1st ed. New York: McGraw-Hill, 2008, pp.293-335.

[10] Forouzan, B., "Cryptographic Hash Functions", *in Introduction to Cryptography and Network Security*, 1st ed. New York: McGraw-Hill, 2008, pp.363-391.

[11] Gilbert, H., and Minier, M., "A collision attack on 7 rounds of Rijndael", *Proceeding of AES Candidate Conference*, New York, USA, April 2000, pp.230-241.

[12] Hong, S., Kim, J., Kim, G., Lee, S., and Preneel, B., "Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192", *Springer-Verlag, proceedings of Fast Software Encryption 12*, Lecture Notes in Computer Science 3557, pp. 368–383, 2005.

[13] Jakimoski, G., and Desmedt, Y., "Related-Key Differential Cryptanalysis of 192-bit Key AES Variants", Springer-Verlag, *proceedings of Selected Areas in Cryptography 2003*, Lecture Notes in Computer Science 3006, pp. 208–221, 2004.

[14] Janadi, A., and Tarah, D., "AES immunity Enhancement against algebraic attacks by using dynamic S-Boxes", *Proceeding of 3rd International Conference on Information and Communication Technologies From Theory to Applications*, Damascus, Syria, 7-11 April 2008, pp. 1-6.

[15] Karpovsky, M., Kulikowski, K., and Taubin, A., "DIFFERENTIAL FAULT ANALYSIS ATTACK RESISTANT ARCHITECTURES FOR THE ADVANCED ENCRYPTION STANDARD", CARDIS, Part 4: Fault Injection Attacks, Toulouse, France, pp. 177-192, 2004.

[16] Kohno, T., "Attacking and Repairing the WinZip Encryption Scheme", *Proceeding of 11th ACM Conference on Computer and Communications Security*, Washington, USA, 25-29 October 2004.

[17] May, L., Henricksen, M., Millan, W., Carter, G., and Dawson, E., "Strengthening the Key Schedule of the AES", *Springer-Verlag Berlin Heidelberg*, ACISP 2002, LNCS 2384, pp. 226–240, 2002.

[18] Osvik, D., Shamir, A., and Tromer, E., "Cache Attacks and Countermeasures: the Case of AES", *Proceeding of RSA Conference Cryptographers Track (CT-RSA)*, LNCS 3860, San Francisco, CA, USA, 8-11 April 2006.

[19] Phan, R., "Impossible Differential Cryptanalysis of 7-round Advanced Encryption Standard (AES)", *Information Processing Letters*, Elsevier, Vol. 91, Number 1, pp. 33-38, 2004.
[20] Schneier, B., "AES Timing Attack", URL: http://www.schneier.com/blog/archives/2005/05/aes_timing_atta_1.html, Retrieved on 17-03-2007.
[21] Smith, W., "1.AES seems weak. 2. Linear time secure cryptography", URL: http://www.math.temple.edu/~wds/homepage/wdsAES.pdf, June 2007.
[22] Sekar, A., Radhika, S., and Anand K., "Secure Communication using 512 Bit Key" European Journal of Scientific Research ISSN 1450-216X Vol.52 No.1 (2011), pp.61-65.
[23] Zhang, W., Wu, W., and Feng, D., "New Results on Impossible Differential Cryptanalysis of Reduced AES", *Springer-Verlag Berlin Heidelberg*, ICISC 2007, LNCS 4817, pp. 239–250, 2007.

Shaaban Sahmoud received the BSc degrees in Computer Engineering in 2006, from the Islamic University of Gaza, Gaza, Palestine. He obtained his Master from Islamic University of Gaza in 2011 in the field of pattern recognition. His research interests include computer vision, security algorithms, image processing, pattern recognition and artificial intelligence. He is a Teaching Assistant at the University Collage of Applied Sciences, Gaza, Palestine.

Wisam Elmasry has received both the BSc and MSc degrees in Computer Engineering from the Islamic University of Gaza (IUG), Palestine in 2004 and 2010 respectively. His research interests include security algorithms and securing MANET routing protocols. He is a lecturer at the Collage of Science and Technology (CST), Khan Younis, Palestine from Septemper 2010 until now.

Shadi Abudalfa has received the BSc and MSc Degrees both in Computer Engineering from the Islamic University of Gaza (IUG), Palestine in 2003 and 2010 respectively. He is a lecturer at the University Collage of Applied Sciences, Palestine. From July 2003 to August 2004, he worked as a research assistant at Projects and Research Lab in IUG. From February 2004 to August 2004, he worked as a teaching assistant at Faculty of Engineering in IUG.