Block Ciphers: The Example of AES

Seminar Report for Research Seminar in Cryptography

Annabell Kuldmaa

Fall, 2014

1 Introduction

Block ciphers are the central tool in the design of protocols for symmetric-key cryptography. A block cipher is be described as a function

$$E: \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n.$$

It means that the function takes two inputs, one being a k-bit string and the other an n-bit string and returns an n-bit string [1]. The first input is called *key* and the second is called *plaintext* and the output is called *ciphertext*. The key-length k and the block-length n are the parameters associated with the block cipher. If the plaintext length is n, then the ciphertext length is n, respectively.

Nowadays most block ciphers are *iterated ciphers.* That means the cipher requires the specification of a round function and a key schedule and the encryption of a plaintext will proceed through a fixed number of similar rounds.

In this paper we give an overview of one of the most commonly used block ciphers – Advanced Encryption Standard. We study the most commonly used attacks on block ciphers and analyze why these attack cannot be applied to AES. Also, we study results of the biclique attack on AES, which is currently the only known attack faster than brute-force on AES.

2 The Advanced Encryption Standard

As described in [1], in 1998 the National Institute of Standards and Technology (NIST) announced a competition for a new block cipher to replace Data Encryption Standard (DES). The relatively short key length of DES was the main problem that lead to the need of replacing DES.

The selection process was notable for its openness and its international flavor. There were three candidate conferences and authors of the candidate ciphers came from 15 different countries [2].

According to [2], AES candidates were evaluated according to three main criteria: security; cost; algorithm and implementation characteristics. Security was essential, any algorithm that was found to be insecure, was not considered further. Cost referred to the computational efficiency of various types of implementation (smart cards, hardware, and software). Algorithm and implementation characteristics included for example flexibility and algorithm simplicity.

As stated in [1], in the end of the selection process, five candidate ciphers out of fifteen were claimed to be secure. Finally, in 2001 NIST announced their choice: an algorithm called Rijndael. Rijndael was designed by two cryptographers from Belgium: Joan Daemen and Vincent Rijmen. The name of the cipher also comes from the names of the authors. Rijndael is descendent of an algorithm called Square.

AES became effective as a federal government standard in 2002. It is also included in the ISO/IEC 18033-3 standard which specifies block ciphers for the purpose of data confidentiality [1].

According to [1], in June 2003, the U.S. government announced that AES could be used to protect classified information, and it soon became the default encryption algorithm for protecting classified information as well as the first publicly accessible and open cipher approved by the National Security Agency (NSA) for top-secret information. AES is one of the Suite B cryptographic algorithms used by NSA's Information Assurance Directorate in technology approved for protecting national security systems.

2.1 Description of AES

The following description is based on [2]. AES is an iterated block cipher which has a block length of 128 bits. It is based on a design principle known as a *substitution permutation network* (SPN) which is a combination of both substitution and permutation. It is also known as confusion and diffusion. There are three permissible key lengths, namely 128 bits, 192 bits and 256 bits. As AES is an iterated block cipher, the number of rounds depends on the key length. The number of rounds (Num) is 10 if the key length is 128 bits, 12 if the key length is 192 bits and 14 if the key length is 256 bits.

In other words, the standard actually specifies three different block ciphers: AES128, AES192 and AES256 but the encryption algorithm for all of them is very similar.

Let χ be the plaintext. The high-level description would be following:

1. Initialize State and perform operation AddRoundKey.
2. For Num-1 rounds, perform a substitution operation called SubBytes, a permutation called ShiftRows, an operation called MixColumns and AddRoundKey.
3. Perform SubBytes, ShiftRows and AddRoundKey.

Define the ciphertext γ be to the State.

2.1.1 State

The plaintext consists of 16 bytes [5]. That means the plaintext can be denoted $\chi = \chi_0, \dots, \chi_{15}$. State is represented by the following 4×4 matrix.

$$\begin{pmatrix} \chi_0 & \chi_4 & \chi_8 & \chi_{12} \\ \chi_1 & \chi_5 & \chi_9 & \chi_{13} \\ \chi_2 & \chi_6 & \chi_{10} & \chi_{14} \\ \chi_3 & \chi_7 & \chi_{11} & \chi_{15} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Figure 1: State representation

2.1.2 AddRoundKey

Operation AddRoundKey means that each byte of the state is combined with a block of the round key using bitwise exclusive or (XOR) [5]. This operation means bitwise addition modulo 2.

2.1.3 SubBytes

In the SubBytes step, each byte $a_{i,j}$ in the state matrix is replaced with a SubByte$(a_{i,j})$ using an 8-bit substitution box called the Rijndael S-box [9]. This operation provides the non-linearity in the cipher. If a block cipher is linear with respect to some field, then, given a few known plaintext-ciphertext pairs, it is possible to recover the key using a simple Gaussian elimination [5]. That means nonlinearity of S-boxes is very important. In contrast to DES S-boxes, the AES S-box can be defined algebraically.

The algebraic formulation of the Rijndael S- boxes includes operations in a finite field GF (*2^8*) which is also known to have good non-linearity properties. Authors of Rijndael stated in [9], that the S-box is constructed by combining the inverse function with an invertible affine transformation, to avoid attacks based on simple algebraic properties.

More specifically, Rijndael S-box takes 8 bits as an input. Converts it to an element of a particular finite field $F_{2^8} = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$ [2]. Multiplication in a finite field is done modulo the irreducible polynomial used to define the finite field. That means in Rijndael multiplication is done modulo $x^8 + x^4 + x^3 + x + 1$ (two elements of the field are multiplied and divided by $x^8 + x^4 + x^3 + x + 1$).

If the input byte is $a_{i,j} = a_7a_6a_5a_4a_3a_2a_1a_0$, then as polynomial it would be described as $a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$. Next step is to find the multiplicative inverse element and then again convert polynomial back to binary. In this finite field all non-zero elements have multiplicative inverse elements.

The last step is to apply a specific affine transformation. SubBytes algorithm is described in the following figure [2]. Function $binaryToField(a)$ converts byte to polynomial, $findInverse(z)$ finds inverse element for a polynomial z and $fieldToBinary(z)$ converts polynomial to binary.

```
$SubBytes(a)\{$
    $z \leftarrow binaryToField(a)$
    $if\ z \neq 0\ then\ z \leftarrow findInverse(z)$
    $a \leftarrow fieldToBinary(z)$
    $c \leftarrow 01100011$
    $for\ i = 0\ to\ 7\{$
        $b_i \leftarrow a_i \oplus a_{i+4} \oplus a_{i+5} \oplus a_{i+6} \oplus a_{i+7} \oplus c_i$
    $\}$
    $return\ b$
$\}$
```

Figure 2. SubBytes Algorithm

2.1.4 ShiftRows

The ShiftRows step operates on the rows of the state and basically is the transposition of the elements of the matrix [4]. It cyclically shifts the bytes in each row by a certain offset.

The first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively [5]. In other words, row n is shifted left circular by n-1 bytes.

Operation ShiftRows in described in the following figure:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,1} & a_{1,2} & a_{1,3} & a_{1,0} \\ a_{2,2} & a_{2,3} & a_{2,0} & a_{2,1} \\ a_{3,3} & a_{3,0} & a_{3,1} & a_{3,2} \end{pmatrix}$$

Figure 3. Operation ShiftRows

In this way, each column of the output state of the ShiftRows step is composed of the bytes from each column of the input state, because of the fact that input block is written column-wise.

2.1.5 MixColumns

As described in [1], in the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes.

ShiftRows and MixColumns provide diffusion in the cipher. The MixColumns step can be viewed as a multiplication by the shown particular MDS matrix (Maximum Distance Separable) in the finite field defined above [4].

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} a_{0,i} \\ a_{1,i} \\ a_{2,i} \\ a_{3,i} \end{pmatrix} \longrightarrow \begin{pmatrix} b_{0,i} \\ b_{1,i} \\ b_{2,i} \\ b_{3,i} \end{pmatrix}$$

Figure 3. Operation MixColumns

Elements of the MDS matrix are elements of the field. Meaning, 01 is 00000001 in binary, therefore 1 as a polynomial. Accordingly, 02 is 00000010 in binary, therefore x as a polynomial and 03 is 00000011 in binary, therefore $x + 1$ as a polynomial.

An equivalent way to explain this step is, to say that we are multiplying $a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0$ by the fixed polynomial $c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{01\}$ and taking the result modulo $x^4 + 1$. [4] a_i denote the elements of a column.

3 Cryptanalysis of Block Ciphers

When it comes to cipher, there is no way around the security of a cipher. In general, there are two most common methods of cryptanalysis of block ciphers: *linear cryptanalysis* and *differential cryptanalysis*. The following paragraphs will summarize the basics of these attacks.

3.1 Linear Cryptanalysis

As described in [2], linear cryptanalysis is a known-plaintext attack. That means an attacker has access to plaintext-ciphertext pairs which are encrypted with an unknown key. In general, the idea of linear cryptanalysis is to find affine approximations to the action of a cipher. The discovery of linear cryptanalysis is attributed to Mitsuru Matsui in 1993.

The attack can be described as follows [3]. In general, linear attack can be divided into two parts: finding linear approximations and deriving key bits. The first step would be constructing linear equations relating plaintext, ciphertext and key bits that have a high *bias*. Bias is the amount by which probability of a linear expression holding deviates from $\frac{1}{2}$. This procedure is different for each cipher. In the most basic type of block ciphers which are based on SPN, the analysis is mainly focused on the S-boxes.

For small enough S-boxes, it is possible to enumerate every possible linear equation relating the S-box's input and output bits, calculate their biases and choose the ones with the highest bias. To get linear approximations for the entire cipher, these results must be combined with other actions of a cipher too. For example, the transposition of the key bits and key mixing.

During this step an attacker will try to obtain a linear approximation of the following form:

$$P_{i1} \oplus P_{i2} \oplus \dots \oplus C_{i1} \oplus C_{i2} \oplus \dots = K_{i1} \oplus K_{i2} \oplus \dots$$

In the equation above P_i denote plaintext bits, C_i ciphertext bits and K_i key bits.

The second step is, to use known plaintext-ciphertext pairs to guess at the values of the key bits involved in the approximation. These key bits can be also called the partial key. We basically apply Matsui algorithm.

In [2], it is stated that for each set of values of the key bits involved in the equation, we count how many times the approximation holds true over all the known plaintext-ciphertext pairs. The partial key whose count has the greatest absolute difference from half the number of plaintext-ciphertext pairs is designated to be the correct one. Because it is assumed that the correct partial key will cause the approximation to hold with a high bias. The magnitude of the bias is significant here, as opposed to the magnitude of the probability itself.

This procedure can be repeated with other linear approximations, obtaining guesses at values of key bits, until the number of unknown key bits is low enough that they can be attacked with brute force [3].

3.2 Differential Cryptanalysis

As described in [2], differential cryptanalysis is a chosen-plaintext attack. That means an attacker can choose plaintexts to be encrypted and obtain the corresponding ciphertexts. Differential cryptanalysis is similar to linear cryptanalysis. Main difference is that differential cryptanalysis involves comparing XOR of two inputs to XOR of the two outputs.

The discovery of differential cryptanalysis is generally attributed to Eli Biham and Adi Shamir in the late 1980s, who published a number of attacks against various block ciphers and hash functions, including a theoretical weakness in the Data Encryption Standard (DES) [10]. It was noted by Biham and Shamir that DES is surprisingly resistant to differential cryptanalysis.

Surprisingly, in 1994, a member of the original IBM DES team, Don Coppersmith, published a paper stating that differential cryptanalysis was known to IBM as early as 1974, and that defending against differential cryptanalysis had been a design goal [10]. According to author Steven Levy, IBM had discovered differential cryptanalysis on its own, and the NSA was apparently well aware of the technique. While DES was designed with resistance to differential cryptanalysis in mind, other contemporary ciphers proved to be vulnerable [10].

As linear cryptanalysis differential cryptanalysis can be also divided into two parts: finding the characteristic and extracting key bits.

In differential attack, the first step is to find a characteristic called differential which relates an input difference to a (Num-1)-st round difference with a non-trivial probability [2]. In other words, the highest probability.

The differential is a tuple $(\Delta X, \Delta Y)$ where $\Delta X = X' \oplus X''$, $\Delta Y = Y' \oplus Y''$, X', X'' denote plaintexts (inputs) and Y', Y''corresponding ciphertexts (outputs).

As stated in [3], this differential choosing is very important. As with linear cryptanalysis, to construct highly likely differential characteristics, the properties of individual S-boxes are examined and used to determine the complete differential characteristic.

When an attacker has found a suitable differential, the next step is to assume that the characteristic holds for the number of rounds minus one (Num-1). Then attacker deduces which round keys for the final round are possible, assuming the difference between the blocks before the final round is fixed.

When one round key has been deemed a potential round key considerably more often than any other key, it is assumed to be the correct round key.

The remaining key bits can be attacked either by brute force or by differential cryptanalysis on Num-1 rounds.

3.3 Wide Trail Strategy of AES

As it might be predicted, AES is secure against these most common attacks on block ciphers (linear and differential cryptanalysis). Furthermore, one of the main cryptographic criterions in the design of Rijndael was its resistance against differential and linear cryptanalysis [8]. The security is referred to a feature called wide trail design strategy. Currently, AES is the best known example of this design.

As described above, differential cryptanalysis exploits differential trails (differentials) with high probability and linear cryptanalysis exploits linear trails (linear approximation) with high correlations (bias). That means both of these trails have in common that they are structures that propagate over multiple rounds. The wide trails strategy aims to avoid the possibility of finding these structures [8].

It is stated in [6], that in the wide trail strategy, the round transformations are composed of two invertible steps. One of them is a local non-linear transformation. By local it is meant that any output bit depends on only a limited number of input bits and that neighboring output bits depend on neighboring input bits. The other one is a linear mixing transformation providing high diffusion.

The structure of the Rijndael round function imposes strict upper limits to the correlation and probability of multiple-round trails. For AES the wide trail strategy is based on the 4×4 MDS matrix over $F_{2^8} = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$ used in MixColumns step [8].

This MDS property is used to ensure that the number of active S-boxes involved in differential and linear attack increases rapidly [6]. That means it is impossible to find differential and linear attacks that involve only few *active S-boxes*.

It is described in [6], that an active S-box transforms a non-zero input difference to a non-zero output difference with some differential probability. Since S-box is also a nonlinear transformation function, the differential probability is not one in general. This means that an active S-box always reduces the probability of differential characteristics. So, one of the security measures against differential cryptanalysis is based on the minimum number of active S-boxes in the cipher which shows the upper bound of differential characteristic probability.

For the AES, the number of active S-boxes in a four- round differential trail is lower bounded by 25. Since the difference propagation probability over an active S-box is at most 2−6, the probability of an 8-round differential trail is below 2−300 [8]. A similar reasoning applies to the case of linear cryptanalysis, where it can be shown that the amplitude of the correlation contribution of a linear 8-round trail is below 2−150 [8].

3.4 Biclique attack on AES

When it comes to cryptanalysis of AES, currently the only known attack faster that brute-force is the biclique attack. Also, unlike previous attacks, the biclique attack does not involve related keys. The following overview is based on a paper [7] by Bogdanov, Khovratovichand and Rechberger. The biclique attack was first developed for hash functions. As differential cryptanalysis, which was first developed for block ciphers, was carried out to hash functions, cryptanalysts were looking for the opposite. Meaning a hash function analysis that could give results on block ciphers.

In general, *meet-in-the-middle attacks* on AES have got less attention than differential and linear cryptanalysis. Meet-in-the-middle attack is a known attack that can exponentially reduce the number of brute force permutations required to decrypt text that has been encrypted by more than one key [11]. The name for this exploit comes from the method, because the attacker tries to break the two-part encryption method from both sides simultaneously, a successful effort enables him to meet in the middle of the block cipher [11]. The reason is that when it comes ciphers with non-linear key schedule and as AES as nonlinear key schedule, the number of rounds broken is rather small. But as stated in [7], meet-in-the-middle attacks with *bicliques* have great potential.

A biclique is characterized by its length (number of rounds covered) and dimension. The dimension is related to the cardinality of the biclique elements and is one of the factors that determines the advantage over brute-force.

Taking the biclique properties into account, there are to different approaches for the key-recovery attack. Suppose that the cipher admits the basic meet-in-the-middle attack on m (out of r) rounds. The first paradigm, called the *long biclique*, aims to construct a biclique for the remaining rounds (meaning r-m). Though the dimension of the biclique decreases as r grows, small-dimension bicliques can be constructed with numerous tools and methods from differential cryptanalysis of block ciphers and hash functions.

The second paradigm, called the *independent biclique*, aims to construct bicliques of high dimension for smaller $b < r - m$ number of rounds efficiently and cover remaining rounds with a new method of matching with precomputations.

The biclique cryptanalysis successfully applies to all full versions of AES and compared to brute-force provides an advantage of about a factor 3 to 5, depending on the version.

Also, it yields advantages of up to factor 15 for the key recovery of round-reduced AES variants with numbers of rounds higher than those cryptanalyzed before. The attacks with lowest computational complexities follow the paradigm of independent bicliques.

The overview of the results can be found in Table 1.

Rounds	Data (bits)	Computations/Suc.Rate	Memory (bits)	Biqlique length in rounds
AES128				
8	$2^{126.33}$	$2^{124.97}$	2^{102}	5
8	2^{127}	$2^{125.64}$	2^{32}	5
8	2^{88}	$2^{125.34}$	2^{8}	3
10	2^{88}	$2^{126.18}$	2^{8}	3
AES192				
9	2^{80}	$2^{188.8}$	2^{8}	4
12	2^{80}	$2^{189.74}$	2^{8}	4
AES256				
9	2^{120}	$2^{253.1}$	2^{8}	6
9	2^{120}	$2^{251.92}$	2^{8}	4
14	2^{40}	$2^{254.42}$	2^{8}	4

Table 1. Results of Biclique Attack on AES [7]

Conclusion

In this report we have studied the concept of block ciphers given the example of AES. As AES is one of the most widely used block ciphers, its security is essential. Currently AES is claimed to be safe, because the biclique attack is still a theoretical attack and all other attacks are not faster than brute-force. I believe that currently we can trust the security of AES, but we still have to keep in mind, that there will probably be a day when AES is broken and needs to be replaces. The question is, how long will it take?

References

[1] M. Bellare. *Block Ciphers*. http://cseweb.ucsd.edu/~mihir/cse107/w-bc.pdf. [Accessed 13 December 2014]

[2] D. R. Stinson. *Cryptography Theory and Practice*. Chapman&Hall/CRC, pp. 85-94, 102-108 , 2002

[3] H.M. Heys. *A Tutorial on Linear and Differential Cryptanalysis*. http://www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.pdf, 2001. [Accessed 13 December 2014]

[4] *Announcing the Advanced Encryption Standard.* http://csrc.nist.gov/ publications/fips/fips197/fips-197.pdf, 2001. [Accessed 13 December 2014]

[5] A. Kak. *Lecture 8: AES: The Advanced Encryption Standard. Lecture Notes on "Computer and Network Security".* https://engineering.purdue.edu/kak/compsec /NewLectures/Lecture8.pdf, 2014. [Accessed 13 December 2014]

[6] S. Cid, S. Murphy and M. Robshaw. *Algebraic Aspects of the Advanced Encryption Standard*. http://www.isg.rhul.ac.uk/~sean/aes_casc04.pdf, 2004. [Accessed 13 December 2014]

[7] A. Bogdanov, D. Khovratovichand C.Rechberger. *Biclique Cryptanalysis of the Full AES.* http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf, 2011. [Accessed 13 December 2014]

[8] J. Daemen, V. Rijmen. *Security of a Wide Trail Design (Progress in Cryptology INDOCRYPT 2002).* http://download.springer.com/static/pdf/102/bok%253A978-3-540-362319.pdf?auth66=1416414830_ae70ac1fd0dcffaea34fd3171412c2bd& ext=.pdf, 2002. [Accessed 13 December 2014]

[9] J. Daemen, V. Rijmen. *AES Proposal: Rijndael*. http://csrc.nist.gov/archive/aes/ rijndael/Rijndael-ammended.pdf#page=1, 2003. [Accessed 13 December 2014]

[10] *Differential Cryptanlysis.* https://www.princeton.edu/~achaney/tmve/wiki100 k/docs/Differential_cryptanalysis.html. [Accessed 13 December 2014]

[11] *Meet-in-the-middle attack.* http://searchsecurity.techtarget.com/definition/ meet-in-the-middle-attack. [Accessed 13 December 2014]