

A differential cryptanalysis of Baby Rijndael

by

Jonathan Wrolstad

A Creative Component submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Information Assurance (Department of Mathematics)

Program of Study Committee:
Clifford Bergman, Major Professor
Douglas Jacobson
Ryan Martin

Iowa State University

Ames, Iowa

2009

Copyright © Jonathan Wrolstad, 2009. All rights reserved.

DEDICATION

I would like to dedicate this Creative Component to my wife Danielle for her loving patience and support.

TABLE OF CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vi
ABSTRACT	vii
CHAPTER 1. Introduction	1
1.1 Ciphers and Cryptosystems	1
CHAPTER 2. Rijndael - AES - Advanced Encryption Standard	5
2.1 AES	5
2.2 Rijndael Structure	6
2.3 Algebra Definitions	7
2.4 Rijndael and $GF(2^8)$	9
CHAPTER 3. Baby Rijndael	13
3.1 Baby Rijndael Structure	13
3.1.1 Introduction	13
3.1.2 The Cipher	14
3.2 Baby Rijndael S-box Structure	16
3.3 Examples	18
3.3.1 Example of Key Expansion	18
3.3.2 Example of Encryption	18
CHAPTER 4. Differential Cryptanalysis	20
4.1 Differential Cryptanalysis Overview	20
4.2 Differential Cryptanalysis of Baby Rijndael S-Box	20

4.3	Differential Cryptanalysis of t Matrix	21
4.4	Differential Cryptanalysis of σ	21
4.5	Characteristic	22
4.5.1	Examples of Characteristics	22
CHAPTER 5. Two, Three and Four Round Differential Cryptanalysis of		
	Baby Rijndael	24
5.1	Two Round Differential Cryptanalysis	24
5.2	Three Round Differential Cryptanalysis	26
5.3	Four Round Differential Cryptanalysis	26
5.4	Recovering the Other Round Keys	29
5.5	Computation (CPU) Time Considerations	29
5.6	Conclusions	30
	BIBLIOGRAPHY	31

LIST OF TABLES

3.1	S-box table lookup.	15
3.2	Different representations for $GF(2^4)$ elements.	17
3.3	The inverse elements.	17
4.1	Differential cryptanalysis of S-Box	21

LIST OF FIGURES

1.1	Iterative block cipher with three rounds (2) pg. 25.	3
1.2	Key-alternating block cipher with two rounds (2) pg. 26.	4
2.1	The matrices A and K	6
2.2	The M matrix. (Where $02 = (00000010)^T$, etc.).	7
2.3	The affine transformation.	10
2.4	SubBytes acts on the individual bytes of the state (1) pg. 16.	10
2.5	ShiftRows operates on the rows of the state (1) pg. 17.	11
2.6	MixColumns operates on the columns of the state (1) pg. 18.	11
2.7	AddRoundKey XOR each column of the state with word from the key schedule. (l is the round number) (1) pg. 19.	12
3.1	SubBytes operation.	15
3.2	ShiftRows operation.	15
3.3	t Matrix	16
3.4	The affine transformation for Baby Rijndael.	16
3.5	Example of key expansion	18
3.6	Example of encryption	19
4.1	Baby Rijndael bit flow	23
5.1	Two round Baby Rijndael with differential trail highlighted.	25
5.2	Three round Baby Rijndael with differential trail highlighted.	27
5.3	Four round Baby Rijndael with differential trail highlighted.	28

ABSTRACT

The Advanced Encryption Standard (AES) was designed in such a way as to prevent linear and differential cryptanalysis attacks against it. We examine the resilience of Baby Rijndael, a scaled-down version of AES with the same algebraic structure as AES, against differential cryptanalysis.

CHAPTER 1. Introduction

First we define some basic concepts in cryptography.

1.1 Ciphers and Cryptosystems

Definition 1.1.1 A *cryptosystem* is a five-tuple $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$, where the following conditions are satisfied:

1. \mathcal{P} is a finite set of possible plaintexts;
2. \mathcal{C} is a finite set of possible ciphertexts;
3. \mathcal{K} , the keyspace, is a finite set of possible keys;
4. For each $k \in \mathcal{K}$, there is an encryption function $e_k \in \mathcal{E}$ and a corresponding decryption function $d_k \in \mathcal{D}$. Each $e_k : \mathcal{P} \rightarrow \mathcal{C}$ and $d_k : \mathcal{C} \rightarrow \mathcal{P}$ are functions such that $d_k(e_k(x)) = x$ for every plaintext $x \in \mathcal{P}$ (3).

The function e_k is called a cipher. There are many different kinds of ciphers. They all take a message as an input and give back some output. The message can be represented in many ways; it may be just an array of letters or words, or it might be text represented as numbers, or even binary numbers. The output also can be some array of letters or numbers. We will call the input plaintext and the output ciphertext. The operation of transforming plaintext into ciphertext is called encryption. The operation of transforming ciphertext into plaintext is called decryption. Suppose Alice wants to send a secret message to Bob. She wants to be sure that nobody except Bob can read the message. This is easy to do if they have some cipher

that nobody but they know or if they share some secret key. However, in many cases, Bob will never see or speak to Alice, so they won't be able to agree upon such a cipher or a key. There are two different kinds of ciphers using keys: public-key ciphers and private-key ciphers. The big difference between these two is that in a private-key cipher, only Alice and Bob know the secret key. In public-key cipher, the key is not secret—everyone may know it. We can define these concepts more precisely as follows.

Definition 1.1.2 A *public-key cryptosystem* is a cryptosystem in which each participant has a public key and a private key. It should be infeasible to determine the private key from knowledge of the public key. To send a message to Alice, Bob will use her public key. Nobody except Alice knows her private key, and one must know the private key to decrypt the message. The most well-known public key cryptosystem is the RSA cryptosystem.

Definition 1.1.3 A *symmetric-key cryptosystem* (or a *private-key cryptosystem*) is a cryptosystem in which the participants share a secret key. To encrypt a message, Bob will use this key. To decrypt the message Alice will either use the same key or will derive the decryption key from the secret key. In a symmetric-key cryptosystem, exposure of the private key makes the system insecure.

In this paper we will be dealing with AES, also called Rijndael, which is a private-key cryptosystem.

Definition 1.1.4 A *block cipher* is a function which maps n -bit plaintext blocks to n -bit ciphertext blocks. The function is parameterized by a key. n is called the block size.

Definition 1.1.5 An *iterated block cipher* (see Figure 1.1) is one that encrypts a plaintext block by a process that has several rounds. In each round, the same transformation or round function is applied to the intermediate result, called the state, using a round key. The set of round keys is usually derived from the user-provided secret key by a key schedule. The number of rounds in an iterated cipher depends on the desired security level. In most cases, an increased number of rounds will improve the security offered by a block cipher (2).

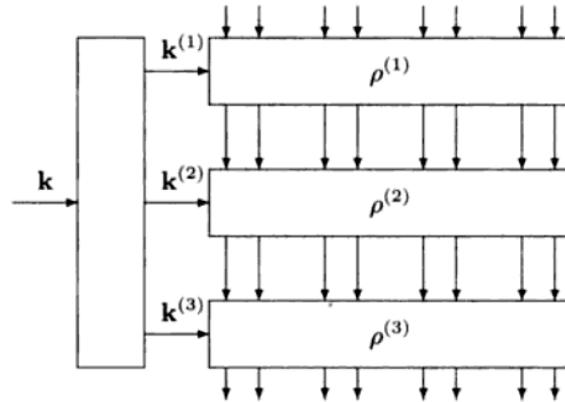


Figure 1.1 Iterative block cipher with three rounds (2) pg. 25.

Definition 1.1.6 A *key-alternating block cipher* is an iterative block cipher with the following properties (see Figure 1.2):

1. *Alternation:* The cipher is defined as the alternated application of key-independent round transformations and key additions. The first round key is added before the first round and the last round key is added after the last round.
2. *Simple key addition:* The round keys are added to the state by means of a simple addition modulo two, called XOR (\oplus) (2).

We said that a block cipher is a function and this is true for any cipher. First, we want to encrypt plaintext, and then we want to decrypt ciphertext to get our plaintext back. So the important condition for our function is to be one-to-one.

Definition 1.1.7 A function is *one-to-one* if no two different elements in the domain are mapped to the same element in image.

The purpose of a cipher is to make communication secure. It is not enough to be sure that you cannot crack the cipher you build, you should be sure that no one else can crack it either. There are many known attacks and you should check your cipher cannot be cracked by any of them, at least not easily. There is one attack that can be applied to any symmetric-key cipher.

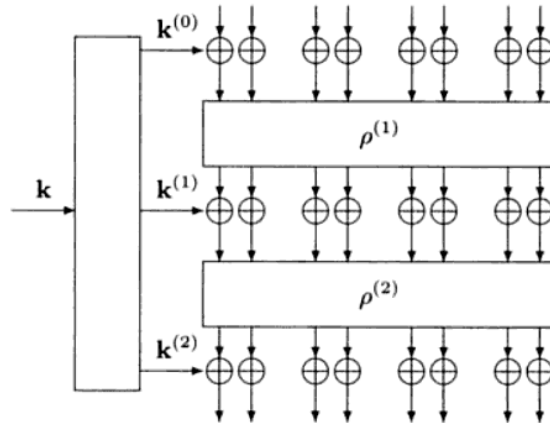


Figure 1.2 Key-alternating block cipher with two rounds (2) pg. 26.

Definition 1.1.8 A *brute-force attack* is a *known-plaintext attack*. It tries every possible key until one of them “works”. That is, knowing the input and output, the attack will try all possible keys until he or she finds a key K so that the encryption of the input with K gives the desired output.

It is easy to see why this attack will always work. The problem is that the running time of such attack might be very large. Even on the fastest computers known this would take centuries. This is why AES can be cracked by brute-force in an idealized world, but based on current known technology, it can not. We should mention that there might be more than one key. That is, if you know both the plaintext and the ciphertext, there might be more than one key that will encrypt plaintext to this ciphertext. The false-positive probability for a key is $1 - e^{-2^{m-n}}$, if we know one block of size n , and the key size is m . If we know two blocks, then the false-positive probability will become smaller: $1 - e^{-2^{m-2n}}$ (6). In our research we will build a new cipher called Baby Rijndael.

CHAPTER 2. Rijndael - AES - Advanced Encryption Standard

2.1 AES

On January 2, 1997, the National Institute of Standards and Technology (NIST) began looking for a replacement for DES – the Data Encryption Standard. DES was used an encryption standard for almost 25 years. The decision was made to replace the standard because the key length of DES was too small and some new attacks using faster new computers could crack DES. Even a Brute-Force search is possible, because for DES, only 2^{55} key options exist. It was not secure anymore. The new cryptosystem would be called AES – the Advanced Encryption Standard. It should have a block length of 128 bits and support key lengths of 128, 192 and 256 bits. Fifteen of the submitted systems met these criteria and were accepted by NIST as candidates for AES. All of the candidaes were evaluated according to three main criteria: security, cost (including computational efficiency), and algorithm and implementation characteristics (such as flexibility and algorithm simplicity). There were five finalists, all of which appeared to be secure. On October 2, 2000, Rijndael was selected to be the Advanced Encryption Standard, because its security, performance, efficiency, implementability and flexibility were judged to be superior to the other finalists (7). Finally, Rijndael was adopted as a standard on November 26, 2001, and published in the Federal Register on December 4, 2001.

There is only one difference between Rijndael and AES. Rijndael can support block and key lengths between 128 and 256 bits which are multiples of 32. AES only has a specific block length of 128 bits and key lengths of 128, 192 and 256 bits.

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \quad K = \begin{pmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{pmatrix}$$

Figure 2.1 The matrices A and K .

2.2 Rijndael Structure

Rijndael is a key iterated and key-alternating block cipher. In this section, we will describe Rijndael with a block size and key size of 128 bits and with 10 rounds. The cipher has the form

$$E(A) = r_{10} \circ r_9 \circ \cdots \circ r_2 \circ r_1(A \oplus K_0).$$

Every round except the last one has the form

$$r_i(A) = (t \circ \sigma \circ S^*(A)) \oplus K_i;$$

where A is the state and K_i is the i th round key. The last round will not have the t operation. We think about A and K as 4×4 arrays of bytes (one byte is equal to 8 bits); see Figure 2.1.

Every round starts by applying S^* to the state A . More precisely, S^* applies a function s to each byte of A . s is a nonlinear, invertible function which takes 8 bits and returns 8 bits. This function is called an S-box and in Rijndael implementation, it is called SubBytes (see Figure 2.4). We will see how this s function works in Section 2.4. For now, we can think about this s function as a table-lookup. We should mention that in some ciphers there is more than one S-box, and different S-boxes are applied on different inputs. In AES the same S-box is applied on all rounds and all inputs.

After computing $S^*(A)$, we apply σ to the result. σ is a permutation function called ShiftRows (see Figure 2.5) and is given by

$$\sigma(a_{i,j}) = a_{i,j-i \bmod 4}.$$

The next function to be applied is t . This function is called MixColumns (see Figure 2.6). $t(A) = M \cdot A$ where M is a 32×4 matrix of bits, see Figure 2.2. t is a linear function and $t(A)$

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 01 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

Figure 2.2 The M matrix. (Where 02 = (00000010)^T, etc.).

is a 4×4 matrix, the new state. At the end of each round, we will XOR the state with the round key. This function is called AddRoundKey (see Figure 2.7).

The only part we still need to explain is how to get the round keys. First, we will construct a list w_0, w_1, \dots, w_{43} , each of which is a 4-byte vector, according to the rule:

$$w_i = \begin{cases} (k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}) & \text{for } i = 0, 1, 2, 3 \\ w_{i-4} \oplus w_{i-1} & \text{for } i \bmod 4 \neq 0 \\ w_{i-4} \oplus S^*(\alpha(w_{i-1})) \oplus c_i & \text{for } i \bmod 4 = 0 \end{cases}$$

α is a function defined by $(x, y, z, u) = (y, z, u, x)$, the c_i 's are constant vectors and k_0, \dots, k_{15} are the bytes of the cipher key. To get K_i for round i , we will just build a matrix with columns $w_{4i}, w_{4i+1}, w_{4i+2}, w_{4i+3}$. In Section 2.4, we will see a more detailed explanation of an S-box. There are more details about Rijndael structure in (2). Rijndael has the interesting property that the decryption algorithm is similar to the encryption algorithm. It will have the same steps, but with inverse functions. In our research we will not use the decryption algorithm, so we will not describe it here. A description of the decryption algorithm can be found in (2).

2.3 Algebra Definitions

In Section 2.4 we will introduce the way to represent each byte of the input of Rijndael as an element of the field $GF(2^8)$. In order to describe this, we need some basic definitions about fields.

Definition 2.3.1 A **group** is an ordered pair (G, \cdot) , where G is a set and \cdot is a binary operation on G satisfying the following axioms:

1. $((a \cdot b) \cdot c) = (a \cdot (b \cdot c))$, for all $a, b, c \in G$.

2. There exists an identity element e such that for all $a \in G$, $a \cdot e = e \cdot a = a$.
3. For each $a \in G$, there is an inverse element a^{-1} such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

Definition 2.3.2 A group is an **Abelian group** if for every $a, b \in G$, $a \cdot b = b \cdot a$.

Definition 2.3.3 A **field** is a set F together with the binary operations addition (+) and multiplication (\cdot) on F such that:

1. $(F, +)$ is an abelian group with identity 0.
2. $(F - \{0\}, \cdot)$ is an abelian group.
3. For all $a, b, c \in F$, $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

$GF(2^8)$ is a field whose elements are polynomials of degree less than 8, with coefficients 0 or 1. In other words, each element of $GF(2^8)$ can be written as: $a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ where $a_i \in \{0, 1\}$, for $i = 0, \dots, 7$. As we can see from the definition of the field, there are two operations defined on $GF(2^8)$: addition and multiplication. Addition is performed by adding the coefficients for corresponding powers of the polynomials modulo 2. For example,

$$(x^7 + x^5 + x^4 + x^2 + x) + (x^6 + x^5 + x^3 + x^2 + x + 1) = x^7 + x^6 + x^4 + x^3 + 1.$$

Multiplication in $GF(2^8)$ is performed by multiplying the two polynomials together and then reducing this product modulo an irreducible polynomial of degree 8. For example, take irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$, then

$$(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1.$$

Now, take the result mod $m(x)$:

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod x^8 + x^4 + x^3 + x + 1 = x^7 + x^6 + 1.$$

Definition 2.3.4 A polynomial is said to be **irreducible** if it cannot be factored into nontrivial polynomials over the same field. For example, over the finite field $GF(2^3)$, the polynomial

$x^2 + x + 1$ is irreducible. However, the polynomial $x^2 + 1$ is not, because it can be written as $(x + 1)(x + 1) = x^2 + 2x + 1 = x^2 + 1$.

There is an inverse element for every element of the field except for the 0 polynomial. The existence of unique inverse element comes from the fact that the multiplication is done modulo an irreducible polynomial. We will think of the inverse element of 0 as 0 itself. We will use these facts in the next chapter.

2.4 Rijndael and $GF(2^8)$

In this section, we describe how to represent a byte as an element of the finite field $GF(2^8)$. Let $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$ represent bits of a byte. Then the corresponding element of $GF(2^8)$ will be $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$. The addition will just be the addition defined for $GF(2^8)$ and multiplication will be defined modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$. In Section 2.2, we said that an S-box is a table-lookup. However, we can actually represent an S-box as a function s acting on elements from $GF(2^8)$. Assume the input to the S-box is some element $a \in GF(2^8)$. We can find the inverse element of a (call it a^{-1}). This inverse element corresponds to multiplication modulo $m(x)$ in $GF(2^8)$. If the S-box function was just defined as an “inverse” function, then it could be attacked using algebraic manipulations. This is why the S-box uses multiplication and addition. This makes the S-box an affine transformation on the inverse of the input. We denote the S-box by f and say that $c = f(d)$, where f is described in Figure 2.3. We apply f on a^{-1} , so actually $d = a^{-1}$.

Another way of thinking about an affine function f is as a polynomial multiplication by fixed polynomial followed by addition of a constant. So the S-box first finds the inverse element of a and then puts it in the affine transformation. The affine function was chosen in such a way so that for all S-boxes, there are no fixed points and no opposite fixed points. That is, $s(a) \oplus a \neq 00$ and $s(a) \oplus a \neq FF$. Where 00 is the zero polynomial of degree 7, and FF is the polynomial of degree 7 with all coefficients equal to 1.

$$\begin{pmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} d_7 \\ d_6 \\ d_5 \\ d_4 \\ d_3 \\ d_2 \\ d_1 \\ d_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Figure 2.3 The affine transformation.

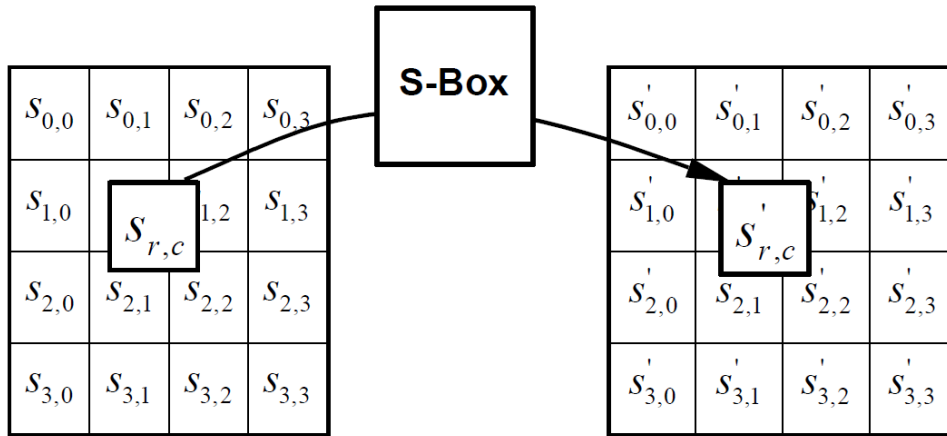


Figure 2.4 SubBytes acts on the individual bytes of the state (1) pg. 16.

The round function of Rijndael has two main parts: the linear part and the non-linear part. The only non-linear part is SubBytes.

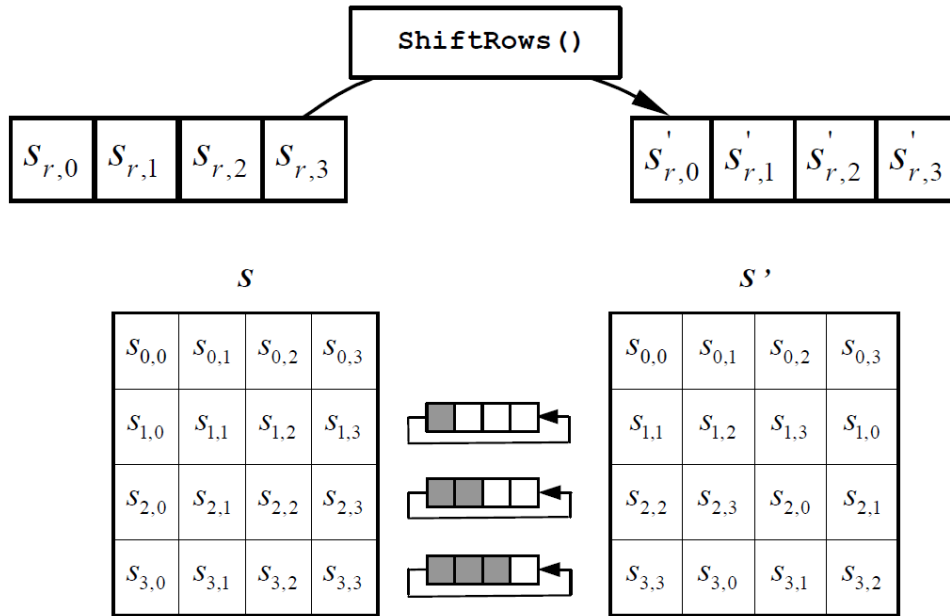


Figure 2.5 ShiftRows operates on the rows of the state (1) pg. 17.

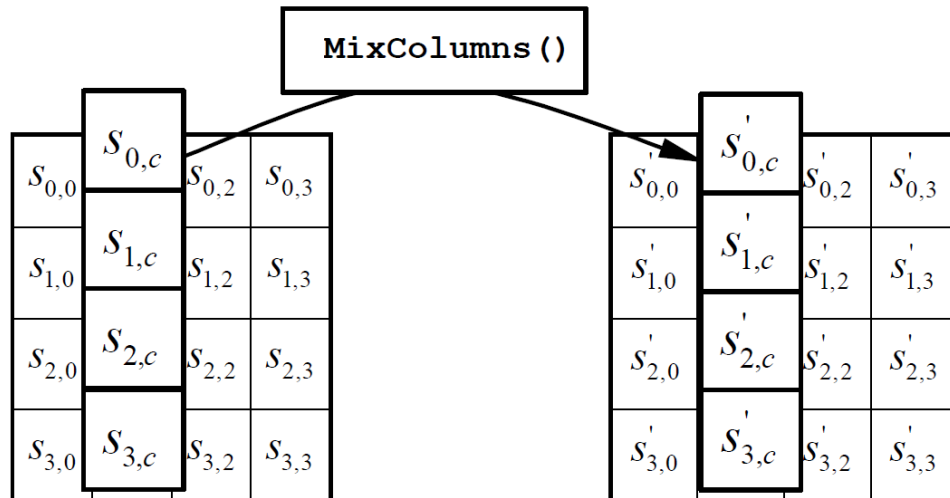


Figure 2.6 MixColumns operates on the columns of the state (1) pg. 18.

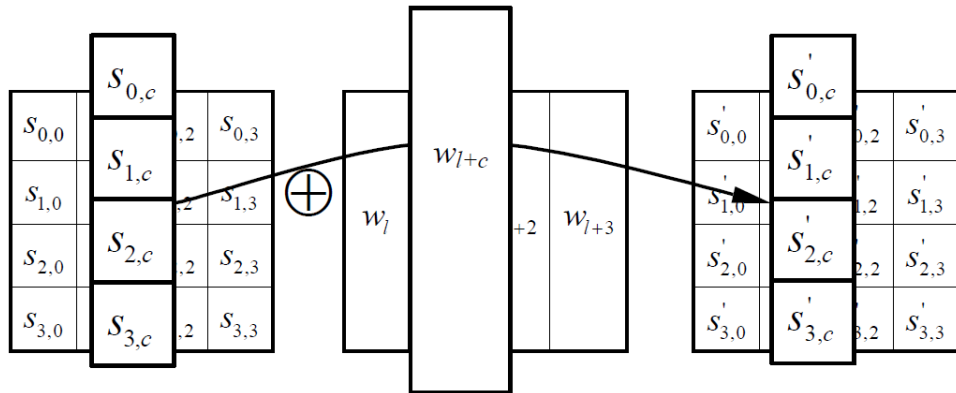


Figure 2.7 AddRoundKey XOR each column of the state with word from the key schedule. (l is the round number) (1) pg. 19.

CHAPTER 3. Baby Rijndael

3.1 Baby Rijndael Structure

3.1.1 Introduction

For the purpose of our research we constructed a new cipher called Baby Rijndael. It is a scaled-down version of the AES cipher. Since Rijndael has an algebraic structure, it is easy to describe this smaller cipher with a similar structure. There were many choices made when Rijndael was constructed, so there would be more than one way to build Baby Rijndael. Baby Rijndael was constructed by Dr. Clifford Bergman. He used this cipher as a homework exercise for the cryptography graduate course at Iowa State University. He wanted a cipher that would help his students learn how to implement Rijndael, but on a smaller, more manageable level. The block size and key size of Baby Rijndael will be 16 bits. We will think of them as 4 hexadecimal digits (called hex digits for short), $h_0h_1h_2h_3$ for blocks and $k_0k_1k_2k_3$ for cipher keys. Note that h_0 consists of the first four bits of the input stream. Note that when h_0 is considered as a hex digit, the first bit is considered the high-order bit. The same is true for the cipher key. For example, the input block 1000 1100 0111 0001 would be represented with $h_0 = 8$, $h_1 = c$, $h_2 = 7$, and $h_3 = 1$. Baby Rijndael consists of several rounds, all of which are identical in structure. The default number of rounds is four, but this number can be changed. Changing the number of rounds is possible and would affect the overall description of the cipher and also the key schedule in a small way. In our attack, we will use Baby Rijndael with two, three and four rounds. The steps of the cipher are applied to the state. The state is usually considered to be a 2×2 array of hex digits. However, for the t operation, the state is considered to be an 8×2 array of bits. In converting between the two, each hex digit is considered to be a

column of 4 bits with the high-order bit at the top. The input block is loaded into the state by mapping $h_0h_1h_2h_3$ to $\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix}$. For example, the input block 1000 1100 0111 0001

would be loaded as $\begin{pmatrix} 8 & 7 \\ c & 1 \end{pmatrix}$ which, as an 8×2 bit matrix is $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$. The state is usually

denoted by a .

3.1.2 The Cipher

At the beginning of the cipher, the input block is loaded into the state as described above and the round keys are computed. The cipher has the overall structure:

$$E(a) = r_4 \circ r_3 \circ r_2 \circ r_1(a \oplus k_0).$$

In this expression, a denotes the state, k_0, k_1, k_2, k_3, k_4 the round keys and

$$r_i(a) = (t \cdot \sigma(s(a))) \oplus k_i.$$

except that in r_4 , multiplication by t is omitted. At the end of the cipher, the state is unloaded into a 16-bit block in the same order in which it was loaded.

Here are descriptions of the individual functions of the cipher.

SubBytes: The s operation is a table lookup applied to each hex digit of the state, as shown in Figure 3.1 where the s function is given by Table 3.1.

In the next section we will describe how to get this table.

ShiftRows: The σ operation simply swaps the entries in the second row of the state; see Figure 3.2.

$$\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix} \xrightarrow{s} \begin{pmatrix} s(h_0) & s(h_2) \\ s(h_1) & s(h_3) \end{pmatrix}$$

Figure 3.1 SubBytes operation.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
s(x)	a	4	3	b	8	e	2	c	5	7	6	f	0	1	9	d

Table 3.1 S-box table lookup.

MixColumns: The matrix t is the 8×8 matrix of bits shown in Figure 3.3. For this transformation, the state is considered to be an 8×2 matrix of bits. The state is multiplied by t on the left using matrix multiplication modulo 2: $a \mapsto t \cdot a$.

KeySchedule: At the beginning of the cipher and at the end of each round, the state is bitwise added (modulo 2) to the round key. The round keys are 2×2 arrays of hex digits similar to the state. The columns of the round keys are defined recursively as follows:

$$w_0 = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix} \quad w_1 = \begin{pmatrix} k_2 \\ k_3 \end{pmatrix}$$

$$w_{2i} = w_{2i-2} \oplus s(\text{reverse}(w_{2i-1})) \oplus y_i \quad w_{2i+1} = w_{2i-1} \oplus w_{2i} \quad \text{for } i = 1, 2, 3, 4.$$

The constants are $y_i = \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix}$ and the “reverse” function interchanges the two entries in the column. Notice that y_i is a vector of length 8 bits and 2^{i-1} and 0 are each four bits. The S function is the same as the one used above. Note that all additions are bitwise modulo 2.

Finally, for $i = 1, 2, 3, 4$, the round key k_i is the matrix whose columns are w_{2i} and w_{2i+1} . Baby Rijndael has a structure similar to Rijndael. If we look at a round of Baby Rijndael, it has the same functions as Rijndael. All the functions constructed for Baby Rijndael were built in the same way as the Rijndael functions, but they will work on a smaller state. For

$$\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix} \xrightarrow{\sigma} \begin{pmatrix} h_0 & h_2 \\ h_3 & h_1 \end{pmatrix}$$

Figure 3.2 ShiftRows operation.

$$t = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Figure 3.3 t Matrix

$$\begin{pmatrix} c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} d_3 \\ d_2 \\ d_1 \\ d_0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Figure 3.4 The affine transformation for Baby Rijndael.

example, ShiftRows for Baby Rijndael works in the same way as ShiftRows of Rijndael works on the 2×2 upper left submatrix of Rijndael. The KeySchedule uses the same definition as in Rijndael, but only for smaller w 's. In the next section we will see how the S-box construction of both ciphers is similar.

3.2 Baby Rijndael S-box Structure

We described in Section 2.4 how to represent a byte as an element of finite field $GF(2^8)$. Now we will show how to represent a hex digit as an element of field $GF(2^4)$. Let $b = b_3b_2b_1b_0$. Then the corresponding element of $GF(2^4)$ will be $b_3x^3 + b_2x^2 + b_1x + b_0$ (see Table 3.2). The addition for $GF(2^4)$ will be defined as usual and multiplication will be defined modulo the irreducible polynomial $m(x) = x^4 + x + 1$. We will not change the structure of the S-box. It will stay same as for Rijndael, but we can not use the same affine transformation f we had before. Now we have a smaller state. Let us define a new f for Baby Rijndael in Figure 3.4.

As in Rijndael, we apply f on a^{-1} , so actually $d = a^{-1}$. Another way of thinking about the affine function f is as polynomial multiplication followed by modulo 2 addition with a constant polynomial. So it can be represented as $s(x) = b(x)g(x) + c(x)$. $g(x)$ is the inverse element

Table 3.2 Different representations for $GF(2^4)$ elements.

hex	binomial	polynomial
0	0000	0
1	0001	1
2	0010	x
3	0011	$x + 1$
4	0100	x^2
5	0101	$x^2 + 1$
6	0110	$x^2 + x$
7	0111	$x^2 + x + 1$
8	1000	x^3
9	1001	$x^3 + 1$
a	1010	$x^3 + x$
b	1011	$x^3 + x + 1$
c	1100	$x^3 + x^2$
d	1101	$x^3 + x^2 + 1$
e	1110	$x^3 + x^2 + x$
f	1111	$x^3 + x^2 + x + 1$

Table 3.3 The inverse elements.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$s^{-1}(x)$	0	1	9	e	d	b	7	6	f	2	c	5	a	4	3	8

of the input to the S-box, $b(x) = x^3 + x^2 + x$ and $c(x) = x^3 + x$. The result will be taken modulo $x^4 + 1$. So the S-box first finds the inverse element of a and then puts it in the affine transformation.

Table 3.3 shows all the inverse elements.

As we can see, our Baby Rijndael S-box has similar properties to the Rijndael S-box. It uses the inverse element and affine transformation and there are no fixed points or opposite fixed points. This is one reason why attacks on Baby Rijndael could give insight into attacks on Rijndael.

$$\begin{aligned}
w_0 &= \begin{pmatrix} 6 \\ b \end{pmatrix} & w_1 &= \begin{pmatrix} 5 \\ d \end{pmatrix} \\
w_1 &\xrightarrow{\text{reverse}} \begin{pmatrix} d \\ 5 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 1 \\ e \end{pmatrix} \oplus w_0 = \begin{pmatrix} 7 \\ 5 \end{pmatrix} \oplus y_1 = \begin{pmatrix} 6 \\ 5 \end{pmatrix} = w_2 & w_1 \oplus w_2 &= \begin{pmatrix} 3 \\ 8 \end{pmatrix} = w_3 \\
w_3 &\xrightarrow{\text{reverse}} \begin{pmatrix} 8 \\ 3 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 5 \\ b \end{pmatrix} \oplus w_2 = \begin{pmatrix} 3 \\ e \end{pmatrix} \oplus y_2 = \begin{pmatrix} 1 \\ e \end{pmatrix} = w_4 & w_3 \oplus w_4 &= \begin{pmatrix} 2 \\ 6 \end{pmatrix} = w_5 \\
w_5 &\xrightarrow{\text{reverse}} \begin{pmatrix} 6 \\ 2 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} 2 \\ 3 \end{pmatrix} \oplus w_4 = \begin{pmatrix} 3 \\ d \end{pmatrix} \oplus y_3 = \begin{pmatrix} 7 \\ d \end{pmatrix} = w_6 & w_5 \oplus w_6 &= \begin{pmatrix} 5 \\ b \end{pmatrix} = w_7 \\
w_7 &\xrightarrow{\text{reverse}} \begin{pmatrix} b \\ 5 \end{pmatrix} \xrightarrow{S} \begin{pmatrix} f \\ e \end{pmatrix} \oplus w_6 = \begin{pmatrix} 8 \\ 3 \end{pmatrix} \oplus y_4 = \begin{pmatrix} 0 \\ 3 \end{pmatrix} = w_8 & w_7 \oplus w_8 &= \begin{pmatrix} 5 \\ 8 \end{pmatrix} = w_9 \\
\text{Where } y_1 &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} & y_2 &= \begin{pmatrix} 2 \\ 0 \end{pmatrix} & y_3 &= \begin{pmatrix} 4 \\ 0 \end{pmatrix} & y_4 &= \begin{pmatrix} 8 \\ 0 \end{pmatrix}
\end{aligned}$$

Figure 3.5 Example of key expansion

3.3 Examples

3.3.1 Example of Key Expansion

Sample key expansion for key= 6b5d.

3.3.2 Example of Encryption

Sample encryption for key= 6b5d and plaintext block= 2ca5. Thus the encryption of 2ca5 under key 6b5d is 6855.

round	start	apply S	apply σ	mult by t	\oplus round key=
input	$\begin{pmatrix} 2 & a \\ c & 5 \end{pmatrix}$				$\oplus \begin{pmatrix} 6 & 5 \\ b & d \end{pmatrix} =$
1	$\begin{pmatrix} 4 & f \\ 7 & 8 \end{pmatrix}$	$\begin{pmatrix} 8 & d \\ c & 5 \end{pmatrix}$	$\begin{pmatrix} 8 & d \\ 5 & c \end{pmatrix}$	$\begin{pmatrix} 2 & f \\ 0 & 7 \end{pmatrix}$	$\oplus \begin{pmatrix} 6 & 3 \\ 5 & 8 \end{pmatrix} =$
2	$\begin{pmatrix} 4 & c \\ 5 & f \end{pmatrix}$	$\begin{pmatrix} 8 & 0 \\ e & d \end{pmatrix}$	$\begin{pmatrix} 8 & 0 \\ d & e \end{pmatrix}$	$\begin{pmatrix} 0 & a \\ e & 3 \end{pmatrix}$	$\oplus \begin{pmatrix} 1 & 2 \\ e & 6 \end{pmatrix} =$
3	$\begin{pmatrix} 1 & 8 \\ 0 & 5 \end{pmatrix}$	$\begin{pmatrix} 4 & 5 \\ a & e \end{pmatrix}$	$\begin{pmatrix} 4 & 5 \\ e & a \end{pmatrix}$	$\begin{pmatrix} d & 9 \\ 2 & 8 \end{pmatrix}$	$\oplus \begin{pmatrix} 7 & 5 \\ d & b \end{pmatrix} =$
4	$\begin{pmatrix} a & c \\ f & 3 \end{pmatrix}$	$\begin{pmatrix} 6 & 0 \\ d & b \end{pmatrix}$	$\begin{pmatrix} 6 & 0 \\ b & d \end{pmatrix}$		$\oplus \begin{pmatrix} 0 & 5 \\ 3 & 8 \end{pmatrix} =$
output	$\begin{pmatrix} 6 & 5 \\ 8 & 5 \end{pmatrix}$				

Figure 3.6 Example of encryption

CHAPTER 4. Differential Cryptanalysis

4.1 Differential Cryptanalysis Overview

Differential cryptanalysis is a chosen-plaintext attack which analyzes the effects of particular differences in plaintext pairs on the differences of the resultant ciphertext pairs. These differences can be used to assign probabilities to the possible keys and to locate the most probable key. This method usually works on many pairs of plaintexts with the same particular difference using the resultant ciphertext pairs (4). For our method the difference (differential) is chosen as a fixed XORed value of the two plaintexts (we will write this difference as x' for plaintexts x and x^* ; similarly we will call ciphertexts and function outputs y and y^* and their difference y'). If we can combine differentials through multiple rounds we call this a differential trail. In this chapter we show how these differences can be used to cryptanalyze Baby Rijndael.

4.2 Differential Cryptanalysis of Baby Rijndael S-Box

The first step in our differential cryptanalysis is to look at the S-box (SubBytes function) of Baby Rijndael. We defined this S-Box in Section 3.1.2. The S-Box takes an input of 4 bits and outputs 4 bits. For now we will call these x and y respectively and write them as $0000 = 0, 0001 = 1, \dots, 1111 = f$. There are 16 possible values for the input and therefore 256 possible values for each input pair. Table 4.1 shows the frequency of certain output differences given a certain input difference. We notice that for each input difference (excluding 0) there are only 7 possible output differences and that one of the output difference occurs 4 out of 16 times. For example, if the input difference is $1 = 0001$ then the output differences 1, 2, 4, 6, 8, and 9 each occur twice and the output difference e occurs 4 times. We will exploit this property in our cryptanalysis.

Table 4.1 Differential cryptanalysis of S-Box

		y'															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x'	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	2	2	0	2	0	2	0	2	2	0	0	0	0	4	0
	2	0	0	2	2	0	0	0	0	2	4	2	0	2	0	0	2
	3	0	4	0	0	2	0	0	2	2	0	2	0	2	2	0	0
	4	0	2	4	0	0	2	2	2	0	0	2	0	0	0	0	2
	5	0	0	0	0	4	0	2	2	0	2	0	2	2	0	0	2
	6	0	0	0	0	0	2	2	0	4	0	2	2	2	0	2	0
	7	0	0	0	2	0	0	4	2	2	0	0	0	0	2	2	2
	8	0	2	0	2	2	2	0	0	2	0	0	2	0	0	0	4
	9	0	2	0	0	0	2	0	0	0	2	0	0	2	4	2	2
	a	0	2	2	2	0	0	2	0	0	0	0	2	4	2	0	0
	b	0	0	2	2	2	4	0	2	0	0	0	0	2	0	2	0
	c	0	0	0	2	2	2	2	0	0	2	4	0	0	2	0	0
	d	0	0	2	0	2	0	0	0	0	0	2	4	0	2	2	2
	e	0	2	0	4	0	0	0	2	0	2	2	2	0	0	2	0
	f	0	0	2	0	0	2	0	4	2	2	0	2	0	2	0	0

4.3 Differential Cryptanalysis of t Matrix

In Section 3.1.2 we defined the t matrix (MixColumns function) for Baby Rijndael. The 8×8 t matrix acts on the 8×2 state matrix. Because the t matrix acts on each column of the state matrix independently we will only analyze how t acts on an 8×1 vector. The first thing to note is that t is an injective linear map. So each of the 256 possible input vectors will get output to all of the 256 possible output vectors with the zero vector getting mapped to the zero vector. Using the linearity of t we also have that for each input difference $x' = x \oplus x^*$ that $t(x \oplus x^*) = t(x) \oplus t(x^*)$. Thus any given input difference will get uniquely mapped to an output difference.

4.4 Differential Cryptanalysis of σ

It is trivial to apply differential cryptanalysis to the σ operation (or ShiftRows function). The σ operation simply takes two 4-bit sections of the state and swaps them. So the differences

of the bits in those 4-bits sections will remain unchanged.

Now we introduce Figure 4.1 which shows the flow of bits in Baby Rijndael and gives names to each bit at each state of encryption. For example, v_{15}^3 is the 15th bit after going through the S-Box in round three.

4.5 Characteristic

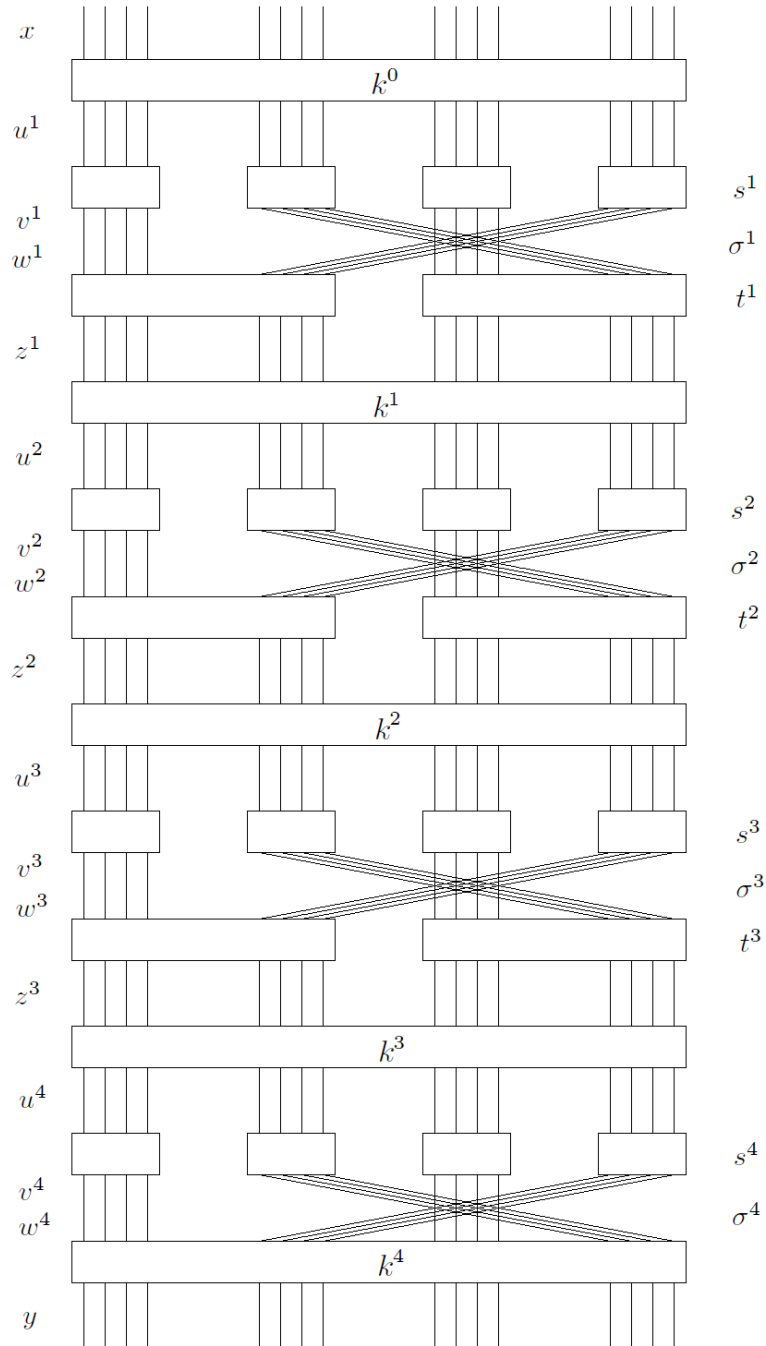
Associated with any pair of encryptions are the XOR values of its two plaintexts, the XOR of its ciphertexts, the XORs of the inputs of each round in the two encryptions and the XORs of the outputs of each round in the two encryptions. These XOR values form an n -round characteristic. A characteristic has a probability, which is the probability that a random pair with the chosen plaintext XOR has the round or ciphertext XORs specified in the characteristic (4).

4.5.1 Examples of Characteristics

Let the XOR of two inputs to Baby Rijndael be $(0, 0, 0, 0)$. Then the one-round, two-round, three-round and four-round characteristics will have the output XOR $(0, 0, 0, 0)$ with probability $p = 1$. This is the trivial case.

For a less trivial example let the XOR of two inputs be $(1, 1, 1, 1)$. Then the one-round characteristic will have an output XOR of $(9, 9, 9, 9)$ with probability $p = \frac{1}{256}$. To compute this first we see that from Table 4.1 the output from each S-Box in round one will be e with probability $p = \frac{1}{4}$. Then of those where the S-box output XOR is e we can say with $p = \frac{1}{16}$ that the XOR input to the t matrix will be (e, e) . The t matrix will give an output XOR of $(9, 9)$ with $p = \frac{1}{16}$. Then because the t matrix acts independently on the other 8 bits in the same manner we finally get a one-round characteristic of $(9, 9, 9, 9)$ with $p = \frac{1}{256}$. It is important to note that $p = \frac{1}{256}$ is much larger than $p = \frac{1}{2^{16}}$ which would be the probability of having a certain XOR of the state just by random chance. We will use these characteristics with higher than usual probabilities to differential cryptanalyze Baby Rijndael.

Figure 4.1 Baby Rijndael bit flow



CHAPTER 5. Two, Three and Four Round Differential Cryptanalysis of Baby Rijndael

Using a differential attack adapted from (3) we are able to recover key bits from two, three and four round Baby Rijndael.

5.1 Two Round Differential Cryptanalysis

First we look at two round Baby Rijndael (see Figure 5.1).

In Chapter 4 we showed how we could find characteristics with probabilities higher than random chance would dictate. Using these higher probabilities we then test possible keys bits for the last round key and the key bits that give the most probable characteristic most often should be the correct key bits. The more plaintext-ciphertext XORs that we use, the higher the probability that our guessed key bits will be correct. Let $x' = (1, 1, 1, 1)$. (Note that $y' = u^{3'}$.) Looking at round one in Figure 5.1 and using Table 4.1, we see the first and fourth S-Box will then each output e with $p = \frac{1}{4}$. We take advantage here of the 4 value in the row corresponding to $x' = 1$ in Table 4.1. The probability of both happening together is $\frac{1}{16}$. So characteristic $z^{1'} = (9, 9, q, q)$ where $q \in \{0, \dots, 15\}$ has $p = \frac{1}{16}$. Then we test all 256 possible combinations of key bits $k_1^2, k_2^2, k_3^2, k_4^2, k_{13}^2, k_{14}^2, k_{15}^2$, and k_{16}^2 . We do this by XORing the possible key bits with y and y^* . We then apply the inverse S-Box to each and XOR them together. This will cancel out k^1 . We then compare the XOR with our most probable characteristic. One particular combination of key bits will make the most probable characteristic occur $\frac{1}{16} \times 256 = 16$ times. These key bits will be the correct key bits. Repeating this process on the middle two hex digits will give their key bits and we will have all of round key k^2 . But we are still missing round

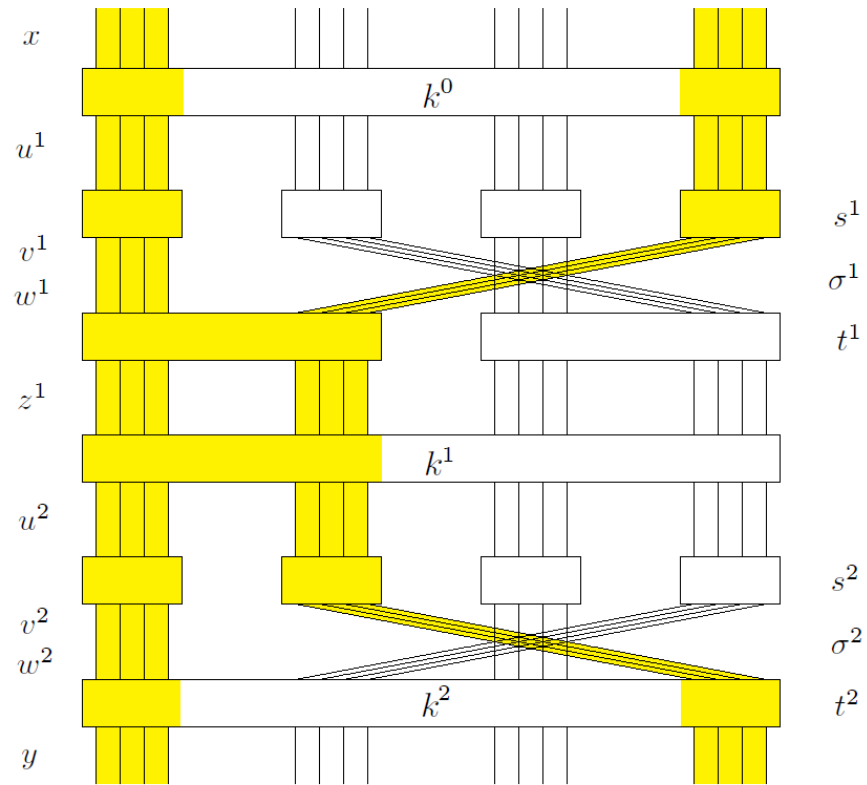


Figure 5.1 Two round Baby Rijndael with differential trail highlighted.

keys k^0 and k^1 . In the last section of this chapter we will discuss how to recover these round keys from k^2 .

5.2 Three Round Differential Cryptanalysis

The three round cryptanalysis is more complicated. See Figure 5.2. In the two round cryptanalysis we were able to only deal with half of the cipher at a time. Now we must use all the input bits in our cryptanalysis. We will use the same method as before on rounds two and three, but we will have to use a new method for round one. Let $x' = (1, 1, 1, 1)$ (note that $y' = u^{4'}$). As we discussed in Chapter 4 this will give a one round characteristic of $(9, 9, 9, 9)$ with $p = \frac{1}{256}$. Once again we take advantage here of the 4 value in the row corresponding to $x' = 1$ in Table 4.1. Starting in round two we will only follow the first and fourth hex digit. Because we are ignoring the second and third hex digit we can move forward with a more likely characteristic which is $(9, q, q, 9)$ where $q \in \{0, \dots, 15\}$. This has $p = \frac{400}{2^{16}}$. We add $\frac{144}{2^{16}}$ to our probability because of other differential trails that end with a 9 in the first and fourth hex digit. Then we apply the same two round method. We get a two round characteristic of $z^{2'} = (2, 2, q, q)$ where $q \in \{0, \dots, 15\}$ with $p = \frac{468}{2^{16}}$. The probability of this characteristic is higher than that of any other. Then just as before in the two round cryptanalysis we test all 256 possible combinations of key bits $k_1^3, k_2^3, k_3^3, k_4^3, k_{13}^3, k_{14}^3, k_{15}^3$, and k_{16}^3 . Using the same method as before we can recover the other hex digits and thus all of the key bits of k^3 .

5.3 Four Round Differential Cryptanalysis

The four round cryptanalysis is by far the most complicated. In the previous two and three round attacks we mainly took advantage of the fact that an S-Box will output a particular difference with $p = \frac{1}{4}$ given a certain nontrivial input difference. But an S-Box also outputs 6 other differences with $p = \frac{1}{8}$. These provide other differential trails that we use to increase the probabilities of our characteristics. Let $x' = (1, 1, 1, 1)$. Computation shows that the two most likely characteristics are $z^{3'} = (14, 6, q, q)$ and $z^{3'} = (q, q, 2, 15)$. Each occur with $p = \frac{336}{2^{16}}$.

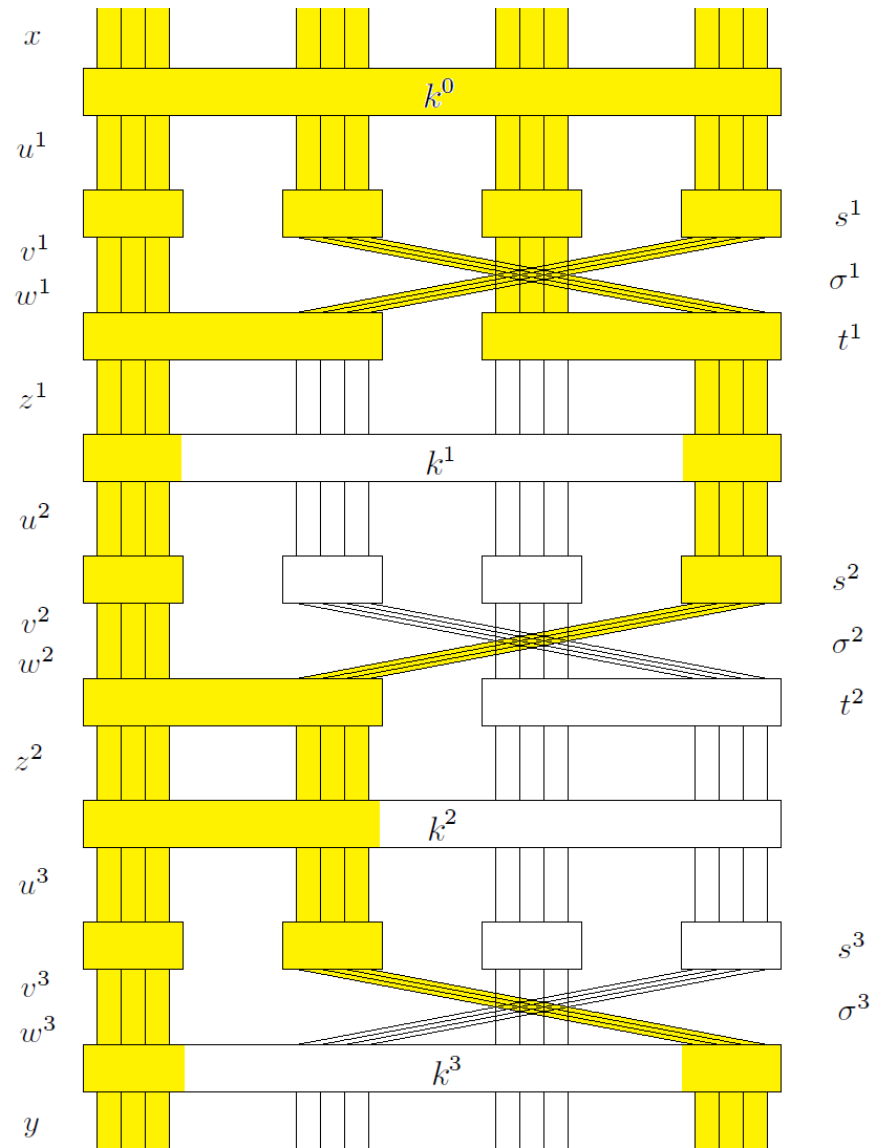


Figure 5.2 Three round Baby Rijndael with differential trail highlighted.

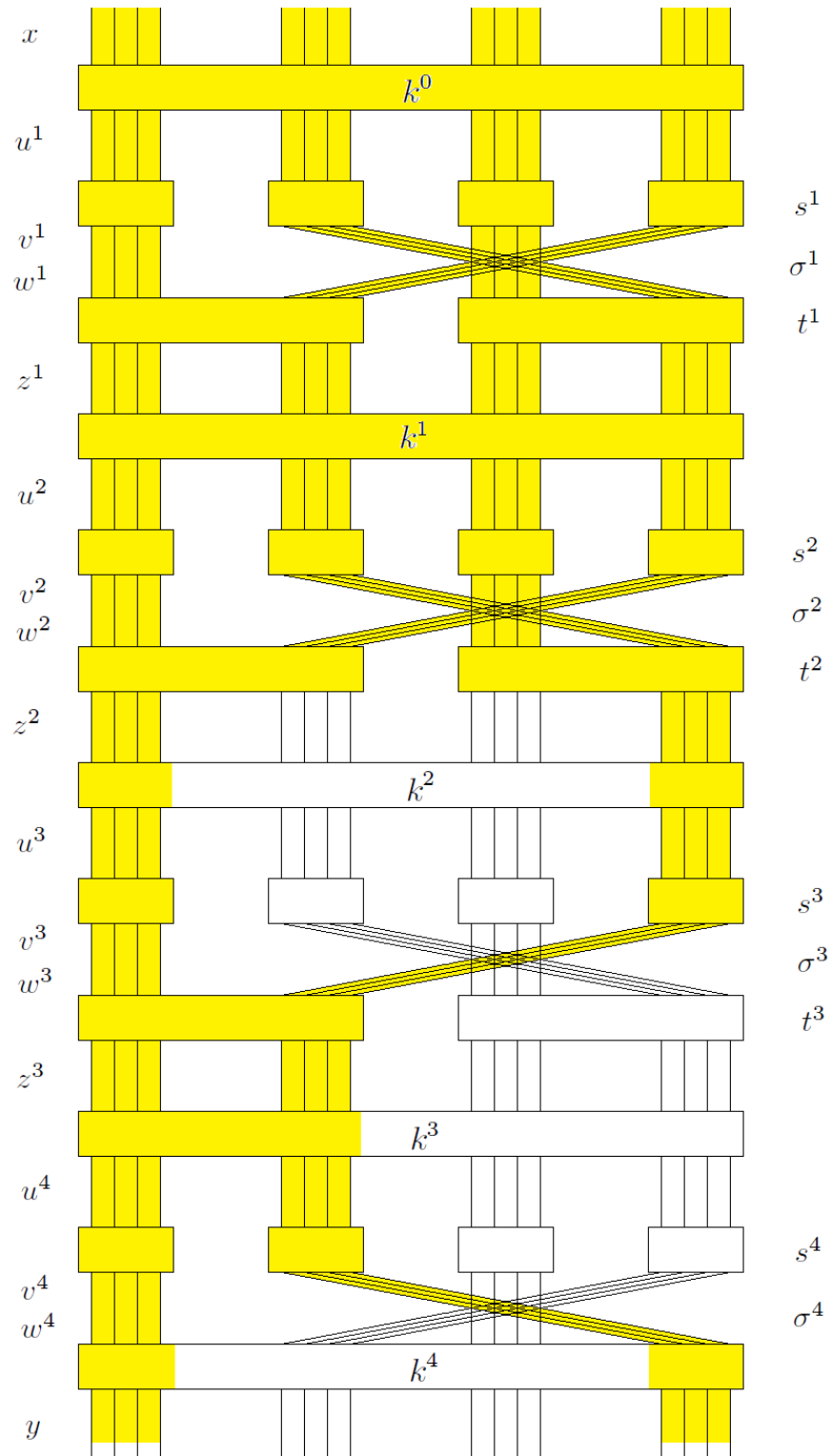


Figure 5.3 Four round Baby Rijndael with differential trail highlighted.

These characteristics can be used in the same way as in the previous two attacks to find the key bits in k^4 .

5.4 Recovering the Other Round Keys

In Chapter 3 the key expansion was defined as follows:

$$w_0 = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix} \quad w_1 = \begin{pmatrix} k_2 \\ k_3 \end{pmatrix}$$

$$w_{2i} = w_{2i-2} \oplus s(\text{reverse}(w_{2i-1})) \oplus y_i \quad w_{2i+1} = w_{2i-1} \oplus w_{2i} \quad \text{for } i = 1, 2, 3, 4.$$

The constants are $y_i = \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix}$ for $i = 1, 2, 3, 4$. First note that $k^0 = w_0w_1$, $k^1 = w_2w_3$, $k^2 = w_4w_5$, $k^3 = w_6w_7$, and $k^4 = w_8w_9$. Fortunately the key expansion works virtually identically in reverse as in the forward direction. We can recover w_{2i-1} from the equation $w_{2i+1} = w_{2i-1} \oplus w_{2i}$ where $w_{2i-1} = w_{2i+1} \oplus w_{2i}$. We can rearrange $w_{2i} = w_{2i-2} \oplus s(\text{reverse}(w_{2i-1})) \oplus y_i$ to be $w_{2i-2} = w_{2i} \oplus s(\text{reverse}(w_{2i-1})) \oplus y_i$. Repeating this process we can get w_0 and w_1 which constitute the original key, k^0 .

5.5 Computation (CPU) Time Considerations

We have demonstrated that it is possible to find the last round key in each of the two, three, and four round attacks. In the previous section we showed how that given the last round key we can compute all of the other round keys and the original key used to generate those round keys. But the ability to recover the key using differential cryptanalysis is only useful if we can recover it faster than we would have recovered the key using a brute force search. It requires only 8.2 seconds on average to do a brute force key search on Baby Rijndael on an Intel Core 2 Quad CPU at 2.66 GHz. Even assuming that each encryption takes the same amount of time (it should take longer) to do a brute force key search on AES with a key size of 128 bits on average it would take 1.35×10^{27} years to recover the key on the same CPU.

So a differential attack is only useful if it takes less encryptions and therefore less CPU time than a brute force key search. Our two round attack takes only 512 encryptions at most and

therefore takes only $\frac{512}{2^{15}} = 1.56\%$ of the time as brute force on average. The three and the four round attacks require almost all of the 2^{16} plaintext-ciphertext XORs. Decreasing the number of plaintext-ciphertext XORs used would decrease the probability of being correct. This makes the three and four round attacks as bad as or even worse than a brute force attack.

5.6 Conclusions

From the two round attack we learned that if Baby Rijndael is cut from four rounds to two rounds it loses its security by a large factor. In the three and the four round attacks the fact that the CPU time is equal to or greater than that required for a brute force search seems to indicate that Baby Rijndael is well designed. And a corollary of this is that AES is well designed. But it has been suggested that a new attack (as yet unknown) could be created based on the fact that AES has a very symmetric structure.

One weakness that I noted was the fact that recovering one round key leads to the discovery of all the round keys. Perhaps a one way function such as a discrete log could somehow be incorporated into the key expansion to prevent this.

BIBLIOGRAPHY

- [1] Advanced Encryption Standard (AES). *Federal Information Processing Standards Publication, 197*. 2001.
- [2] Joan Daemen, Vincent Rijmen. *The Design of Rijndael*. Springer-Verlag, 2002.
- [3] Douglas R. Stinson. *Cryptography Theory and Practice. Third Edition*. Chapman and Hall/CRC, 2006.
- [4] Eli Biham, Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer-Verlag, 1993.
- [5] Elizabeth Kleiman. *The XL and XSL attacks on Baby Rijndael*. Thesis, 2005.
- [6] Dr. Clifford Bergman. *Lecture Notes*. 2009.
- [7] National Institute of Standards and Technology. *Advanced Encryption Standard (AES) Development Effort*. <http://csrc.nist.gov/archive/aes/index.html>. February 28, 2001. Retrieved March 2009.