A ZERO-DIMENSIONAL GROEBNER BASIS OF AES-128 AND ITS CRYPTANALYTIC IMPLICATION

By

Hannibal Lemma

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
ADDIS ABABA UNIVERSITY
ADDIS ABABA, ETHIOPIA
JUNE 2012

ADDIS ABABA UNIVERSITY
DEPARTMENT OF MATHEMATICS

The undersigned hereby certify that they have read and recommend to the School of Graduate Studies for acceptance a thesis entitled "**A Zero-Dimensional Groebner Basis of AES-128 and Its Cryptanalytic Implication**" by **Hannibal Lemma** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: June 2012

Advisor: _____
Birhanu Bekele (Ph.D.)

Examiners: _____
K. Venkateswarlu (Ph.D.)

_____
Zelalem Teshome (Ph.D.)

ADDIS ABABA UNIVERSITY

Date: **June 2012**

Author: **Hannibal Lemma**

Title: **A Zero-Dimensional Groebner Basis of AES-128 and Its Cryptanalytic Implication**

Department: **Mathematics**

Degree: **M.Sc.** Convocation: **June** Year: **2012**

Signature of Author

To my parents and my sister.

Table of Contents

Abstract

After a detail demonstration of AES (Advanced Encryption Standard) , we explained the procedure of changing the key-recovery problem of AES-128 to a zero dimensional ideal from a single plain text /cipher text pair. Followed by explaining a method, Buchberger's Algorithm for computing a Groebner basis for this ideal. Then, finally we will discuss on the implication of the result from the cryptanalytic point of view. (i.e. What security implication the existence of the Groebner basis has?)

Acknowledgements

I would like to thank Dr. Birhanu Bekele, my advisor, for his many suggestions and constant support during this project. Specially, for letting me do things in a way I intended them, and the confidence he has on me.

I am grateful to my parents and my sister for their patience, unconditional love and support. Without them this work would have been just difficult.

Finally, I wish to thank the following: Daniel Workneh, Solomon Bekele and Solomon Belete (for their friendship and technical support); **INSA** (for the introduction to the science of Cryptography); My friends here and abroad (for telling me that 'I can' even I doubted myself).

Addis Ababa, Ethiopia — Hannibal Lemma
June, 2012

Introduction

For a given ideal of ring of polynomial, one can have different number of basis. A *Groebner basis*[6] is mostly preferable, because it has some useful properties that enable us to answer main questions concerning ideals of ring of polynomials of several variables. Two of these questions are the membership problem and solving system of polynomial equations.

Membership problem [6]: given a *Groebner basis* of an ideal I subset of the polynomial ring $R = F[x_1, ..., x_n]$ we can efficiently decide whether the polynomial f lies in I or not.

Solving system of polynomial equation [6] for a suitable term order, the variety of the ideal can be efficiently computed.

There are many ways (algorithms) of computing the *Groebner basis* of a set of polynomials. Some of these algorithms are the *Buchberger Algorithm* [6] and *Faugers F4* [8]. These algorithms involve polynomial reduction. In other words, it is difficult to predict the time and space complexity of these algorithms. For polynomials in large number of variables, these algorithms quickly become infeasible.

Since 2001, *Rijndeal*[11] became a widely used block cipher, after it has been selected as a standard (Advance Encryption Standard) by *NIST (National Institute of Standards and Technologies)*. However, there were some criticisms on *Rijndeal*,

which are the mathematical simplicity and rich algebraic structure. Though, no one was able to change this fact to cryptanalytic attack.

Mathematical simplicity: in *Rijndeal* algorithm (cipher) except the S-box (substitution box)[11] all the steps (functions) are linear.

Nicolas Courtois and Josef Pieprzyk[5] in 2002 have demonstrated how to obtain over-defined system of quadratic equation over GF(2)(finite field of order 2). And, on their own way *Sean Murphy and Matthew J. B. Robshow* in 2002 have constructed an embedding for the AES called *Big Encryption system* (*BES*) for which a system of over defined quadratic equation over $GF(2^8)$ exist for the *Rijndeal* block cipher. Both approaches have limited their multivariate polynomial system of equation to smaller degree (less than or equal to 2). This is because representations of the output of S-box as a polynomial expression are relatively high degree. In which the time and space complexity has direct relation with degree and number of polynomials.

Using these representation we can describe the key recovery problem for AES cipher with a key length of 128 bits as a system of 200 polynomial equations of degree 254 and 152 linear equations.

Johannes Buchman, Andrie Pyshkin and Ralf-Philipp Weinman[3] on their paper *A Zero Dimensional Groebner Basis for AES-128* have explained that how to compute a *Groebner basis* for a zero dimensional ideal, which is describing the key-recovery problem for full *AES-128*.

In this thesis, we will see that how this system of 200 polynomial equations of degree 254 and 152 linear equation can be obtained. Later, by choosing the appropriate term order and by applying some linear operations we will show how to generate a *Groebner basis* for *AES-128* from this system. At last we will describe the

cryptanalytic implication of the obtained *Groebner basis*.

The paper is structured as follows: in the first Chapter we will explain *Rijndeal* cipher in detail, in the second chapter we will explain how the key recovery problem can be represented as a *system of polynomial equations*, in the third chapter the terms, properties and concepts behind *Groebner basis* will be discussed and explain how to construct a *Groebner basis* for *AES-128*. And, finally on the fourth chapter we discuss the efficiency of algorithms (used to convert the *Groebner basis* w.r.t one term order to an other) and their complexity together with the Cryptanalytic implication of the result.

Terms

AES: Advanced Encryption Standard

Affine Transformation: A transformation consisting of multiplication by a matrix followed by addition of a vector.

Array: An enumerated collection of identical entities (e.g. an array of bytes)

Bit: A binary digit having a value of 0 or 1

Block: Sequence of binary bits that compromise the input, output, State, and Round Key. The length of the sequence is the number of bits it contains.

Byte: A group of eight bits that is treated either as a single entity or as an array of 8 individual bits.

Cipher: Series of tramsformation that convert plaintext to ciphertext using Cipher Key

Cipher Key: Secret, cryptographic key that is used by the Key expansion routine to generate a set of Round Keys.

Ciphertext: Data output from the Cipher or input to the Inverse Cipher

Inverse Cipher: Series of transformation that converts ciphertext to pliantext using the Cipher Key

Key Expansion: Routineused to generate a series of Round Key from the Cipher Key

Plaintext: Data input to the Cipher or output or the Inverse Cipher

Rijndeal: name of a Cryptographic Algorithm

Round Key: Round Keys are values derived from the Cipher Key using the Key Expansion routine

State: Intermaediate Cipher result that can be pictured as a rectangular array of bytes.

S-box: Non linear substitution table used in several byte substitution transformation and in the Key Expansion routine to perform a one-for-one substitution of a byte value.

Word: A group of 32 bits that is treated as a single entity or as an array of 4 bytes

Algorithm Parameters, and Functions

AddKey: Transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using the XOR operation.

InvMixColumns: Transformation in the Inverse Cipher that in the inverse of MixColumns.

InvShiftRows: Transformation in the Inverse Cipher that is the inverse of ShiftRows.

InvSubBytes: Transformation in the Inverse Cipher that is the inverse of SubBytes.

K: Cipher Key.

MixColumn: Transformation in the Cipher that takes all of the columns of the

State and mixes their data (independent of one another) to produce new columns.

N_b: Number of columns (32-bit words) comprising the State.

N_k: Number of 32-bit words comprising the Cipher Key.

N_r: Number round, which is a function of N_k and N_b.

Rcon: The round constant word array.

RotWord: Function used in the Key Expansion routine that takes a four-byte word and performs a cyclic permutation.

ShiftRows: Transformation in the Cipher that processes the state by cyclically shifting the last three rows of the State by different offsets.

SubBytes: Transformation in the Cipher that processes the State using a non-linear byte substitution table (S-box) that operates on each of the State bytes independently.

SubWord: Function used in the Key Expansion routine that takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word.

XOR: Exclusive-OR operation.

Binary to hexadecimal representation

Binary	Hexadecimal	Binary	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	a
0011	3	1011	b
0100	4	1100	c
0101	5	1101	d
0110	6	1110	e
0111	7	1111	f

Chapter 1

Preliminaries

1.1 Finite Fields

Definition 1.1.1. *Finite field is a field with finite number of elements.*

For every prime number P and positive integer n, there exists a field of order P^n. Finite fields of order P^n are unique upto-isomorphic. This field is called Galois field of order P^n and denoted by $GF(P^n)$.

Theorem 1.1.2. *:If f is an irreducible polynomial in $GF(P)[x]$ of degree n, then the quotient ring $GF(P)[x]/ < f(x) > \cong GF(P^n)$.*

Example 1.1.3. *$Q(x) = x^8 + x^4 + x^3 + x + 1$ is irreducible in $GF(2)$, because both $Q(1) = Q(0) = 1 \neq 0$. That is Q has no solution in $GF(2)$. Therefore, according to the theorem $GF(2)[x]/\langle Q(x)\rangle \cong GF(2^8)$. We later call this polynomial $Q(x) = x^8 + x^4 + x^3 + x + 1$ Rijndeal polynomial.*

1.1.1 Addition

The addition of two elements in a finite field is achieved by adding the coefficients of the corresponding powers in the polynomials for the two elements. The addition is performed with the XOR operation for our case. That is modulo 2, $1 \oplus 1 = 0, 1 \oplus 0 = 1, 0 \oplus 0 = 0$.

As a result, subtraction of polynomials is identical to addition of polynomials. For our case in the example, addition of finite field elements can be described as *modulo 2* addition of corresponding bits in the byte. For two bytes { a_7, a_6 a_5 a_4 a_3 a_2 a_1 a_0 } and { b_7, b_6 b_5 b_4 b_3 b_2 b_1 b_0 }.

The sum is { c_7, c_6 c_5 c_4 c_3 c_2 c_1 c_0 }, where each $c_i = a_i \oplus b_i$.

For example the following expressions are equivalent to one another.

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ Polynomial notation

$\{01010111\} \oplus \{10000011\} = \{11010100\}$ Binary notation.

By the addition defined before the element {00000000} is the additive identity and every element is its own additive inverse.

1.1.2 Multiplication

In the polynomial representation, multiplication in $GF(2^8)$ corresponds with the multiplication of polynomials modulo an irreducible polynomial of degree 8. In our case the irreducible polynomial is $Q(x) = x^8 + x^4 + x^3 + x + 1$.

Example 1.1.4. { *01010111* } $\otimes$ { *10000011* } = { *11000001* } *in other words:* $(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1)(mod x^8 + x^4 + x^3 + x + 1)$
$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) mod (x^8 + x^4 + x^3 + x + 1)$
$= (x^7 + x^6 + 1)$
= {11000001} *Binary notation*

By multiplication defined above, the element {00000001} is the multiplicative identity. For non-zero elements with polynomial notation $b(x)$ of degree less than 8, the multiplicative inverse of $b(x)$ can be found as follows. Recall the *Extended Euclidean Algorithm*, then there exist $a(x)$ and $c(x)$ such that

$b(x)\ a(x) + Q(x)\ c(x) = 1$, where $Q(x) = x^8 + x^4 + x^3 + x + 1$

Hence $a(x)b(x) mod Q(x) = 1$

$\Rightarrow b^{-1}(x) = a(x) mod Q(x)$.

Moreover, the multiplication is distributing over the addition.

1.2 Interpolation

Interpolating Polynomial; two points determine a straight line. More precisely, any two points in the plane, (x_1,y_1) and (x_2,y_2), with $x_1 \neq x_2$, determine a unique first degree polynomial in x whose graph passes through the two points. There are many different formulas for the polynomial, but they all lead to the same straight line graph. This can be generalized for points more than two. Given n-points in the plane,(x_k,y_k), $k = 1, ..., n$ with distinct x_k's, there is a unique polynomial in x of degree less than n whose graph passes through the points. Observe that n, the number of data points, is also the number of coefficients, although some of the leading coefficients might be zero. So the degree might actually be less than $n - 1$. This polynomial is called the interpolating polynomial because it exactly reproduce the given data. That is $P(x_k)= y_k$ for all $k = 1, ..., n$.

The widely used representation of the interpolating polynomial is the Lagrange form,

$$P(x) = \sum_k (\prod \frac{x - x_j}{x_k - x_j})$$

There are n-terms in the sum and $n - 1$ term in each product, so this expression defines a polynomial of degree at most $n - 1$. If $P(x)$ evaluated at x=x_k all the products except the k^{th} are zero and the k^{th} product is equal to one, so the sum is equal to y_k the interpolation conditions are satisfied.

Example 1.2.1. *Consider the following data (0,-5), (1,-6), (2,-1), (3,16)*

The Lagrange form of the polynomial interpolating these data is:

$$\begin{aligned} P(x) &= \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} * (-5) + \frac{x(x-2)(x-3)}{(1-0)(1-2)(1-3)} * (-6) \\ &+ \frac{x(x-1)(x-3)}{(2-0)(2-1)(2-3)} * (-1) + \frac{x(x-1)(x-2)}{(3-0)(3-1)(3-2)} * (16) \end{aligned}$$

One can easily observe that each term is of degree three, so the entire sum has degree at most three. Because the leading term does not vanish, the degree is actually three. And, if we compute P at $x = 0, 1, 2, 3$ three of the terms vanish and the fourth produce corresponding value from the data set. The interpolating polynomial representing the above data set is $P(x) = x^3 + 2x - 5$.

1.3 Symmetric (Private) Key Encryption

Provide secrecy when two parties (*Alice* and *Bob*) communicate. *Alice* and *Bob* first agree on a key k. That is *Alice* encrypt m *(message)* using encryption algorithm E and the key k and obtain $c = E(k, m)$ and send c to *Bob*. By using the decryption algorithm D and the same key k, *Bob* decrypt c to recover the message $m = D(k, c)$.

Definition 1.3.1. *A Symmetric key encryption scheme consists of a map $E_k : M \to C, m \to E(k, m)$ is invertible.*
The elements $m \in M$ are plaintext ,
C is the set of the ciphertext,
$k \in K$ are the keys,
E_k is called the encryption function with respect to the key k.
The inverse function $D_k = E_k^{-1}$ is called the decryption function.
There are two types of Symmetric key encryption Block and Stream ciphers.

1.3.1 Block Ciphers

Definition 1.3.2. *[7] Block cipher is a Symmetric key encryption scheme with $M = C = \{0,1\}^n$ and $K = \{0,1\}^r$*

$$E_k : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}^n, (k, m) \to (k, m).$$

That is using the secret key k of binary length r and plain text m of length n resulting cipher text $C = E(k, m)$ also have binary length n.

1.4 Cryptanalysis

The term Cryptanalysis has given a lot of meaning by different scholars in different time. We try to see it from its most classic definition to today's widest definition.The word Cryptanalysis is a combination of two Greek words Kryptos and Analyein which individually mean hidden and to untie respectively.

The modern definition of cryptanalysis in the early time of modern cryptology said:

Cryptanalysis is the process of breaking someone else's cryptographic writing. Following the introducing of different methods of cryptanalysis, the definition of cryptanalysis is better to be saying: Cryptanalysis is a complex process involving statistical analysis, analytical reasoning, mathematical tools, and pattern finding. Cryptanalysis is the process of analysing ciphers, cipher text and cryptosystem finding weaknesses in cryptographic algorithms rather than breaking a cryptographic algorithm.

Cryptanalysis is the science of 'code-breaking,' in which a person reconstructs the original plaintext message from an encrypted version. Cryptanalysis is different than simply 'decoding' or 'deciphering' a cryptogram since cryptanalysts do not have prior knowledge of the encryption or key used to encrypt and must determine these things on their own. This is no easy task. Cryptanalysts spend hours, weeks, and even months scribbling on paper, testing hypotheses, and rereading the encrypted message so many times that they're able to repeat it from memory easily. Cryptanalysis is a nerve-racking profession that consumes a person. A difficult cryptogram will sit in one's mind every second of every day and scream for constant attention until it is solved.

It is better to know some highlights about types of cryptanalysis.

1.4.1 Types of Cryptanalysis

There are several distinct types of cryptanalytic attack. The type used depends on the type of cipher and how much information the cryptanalyst has. A standard cryptanalytic attack is to determine the key which maps a known plaintext to a known ciphertext. This plaintext can be known because it is standard or because it is guessed. If the plaintext segment is guessed it is unlikely that its exact position is known however a message is generally short enough for a cryptanalyst to try all possible positions in parallel. In some systems a known ciphertext-plaintext pair will compromise the entire system however a strong encryption algorithm will be unbreakable under this type of attack.

A **Brute Force Attack** requires a large amount of computing power and a large amount of time to run. It consists of trying all possibilities in a logical manner until the correct one is found. For the majority of encryption algorithms a brute force attack is impractical due to the large number of possibilities.

Another type of brute force attack is a dictionary attack. This essentially involves running through a dictionary of words in the hope that the key (or the plaintext) is one of them. This type of attack is often used to determine passwords since people usually use word, that they can easily remember.

In a **Ciphertext Only Attack** the cryptanalyst has only the encoded message from which to determine the plaintext, with no knowledge whatsoever of the actual message. A ciphertext only attack is presumed to be possible, if not easy. In fact, an encryption techniques resistance to a ciphertext only attack is considered the basis for its cryptographic security.

In a **Chosen Plaintext Attack** the cryptanalyst has the capability to find the

ciphertext corresponding to an arbitrary plaintext message of his or her own choosing. The likelihood of this type of attack being possible is not much. Codes which can survive this attack are considered to be very secure.

Kronecker (tensor) Product of two Matrices

For matrices U and V, Kronecker product of U and V, U $\otimes$ V is given by

$$U \otimes V = \begin{bmatrix} u_{11}V & u_{12}V & . & . & . \\ u_{11}V & u_{11}V & . & . & . \\ . & & & & \\ . & & & & \\ . & & & & \end{bmatrix}$$

Example 1.4.1.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{12}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \\ a_{21}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} & a_{22}\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{pmatrix}$$

Chapter 2

Advanced Encryption Standard (AES)

There was world wide invitation of **NIST** (National Institute of Standards and Technologies) in January 1997 for selection of a new encryption standard. In the selection process proposals were required to support a block size of atleast 128-bits and key size of 128, 192 and 256 bits. After two consecutive round of selection *Rijndeal* (Daemen and Rijmen) selected to be ***AES*** (*Advanced Encryption Standard*) by **NIST** in October 2000 [7]

Rijndeal was developed by *J.Deamon* and *V. Rijmen*. It is an iterated block cipher and support different block and key sizes. Block and key size of 128, 160, 192, 224 and 256 bits can be combined independently. The only difference between *Rijndeal* and *AES* is that *AES* only support a subset of *Rijndeal's* block and key size. The *AES* fixes the block length to 128 and uses the three key lengths 128, 192 and 256 bits [7]. For the sake of simplicity we name *AES* as *AES-128*, *AES-192* and *AES-256*, depending on the key size we use.

2.1 Structure of Rijndeal

Rijndeal is an iterated block cipher. The iterations are called *rounds*. The number of rounds, which we denote by N_r, depends on the block length and the key length. In each round except the last round, the same round function is applied, each time with different round key. The round function of the last round differs slightly. The round keys key_1, ..., key_{N_r} are derived from the key K by using the key schedule algorithm, which we will see it later.

Rijndeal is byte oriented. Input and output (*Plaintext, Key, Ciphertext*) are considered as one dimensional array of 8-bit-bytes. And, both block length and key length are multiples of 32-bits. Here N_b and N_k denote the block length in bits divided by 32 and the key length in bits divided by 32. In other words *Rijndeal* block consists of N_b words (or $(4N_b)$bytes) and a *Rijndeal* key consists of N_k words (or $(4N_k)$bytes). The following table shows the number of rounds N_r as a function of N_k and N_b.

N_k	N_b				
	4	5	6	7	8
4	10	11	12	13	14
5	11	11	12	13	14
6	12	12	12	13	14
7	13	13	13	13	14
8	14	14	14	14	14

Particularly, *AES* with key length 128-bits consists of 10-rounds.

The round function of *Rijndeal* and its steps, operate on an intermediate result called the *state*. The state is block of N_b words. At the beginning of the encryption, the variable state is initialized with the plaintext block, and at the end, the state contains the ciphertext block. The state is considered as a 4-row matrix of bytes with

N_b columns. Each column contains one of the N_b words of the state. The next table (matrix) shows the state matrix in case of block length of 128 bits. In this case we have 4 state words. Each column of the matrix represents a state word consists of 4-bytes.

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

2.2 The Rijndeal Algorithm

An encryption with *Rijndeal* consists of initial round key addition, followed by applying the round function (N_r-1)-times, and the last round with a slightly modified round function. The round function is composed of the *SubBytes*, *ShiftRows* and *MixColumns* steps and an addition of the round key. In the final round the *MixColumns* step is omitted.

Encryption Algorithm

byte string Rijndeal (byte string plain text block, key)

Init State (Plain text Block, State)

AddKey(State,key_0)

while (1$\leq$ i $\leq N_{r-1}$) do

SubBytes(State)

ShiftRows(State)

MixColumns(State)

AddKey(State, key_i)

end while; SubBytes(State)

ShiftRows(State)

AddKey(State, key_{N_r})

return State

Now, we see the round function in detail.

The input and output of the *Rijndeal* algorithm are byte strings of 4-N_b bytes. At the beginning, the state matrix is initialized with the plaintext block. The matrix is filled column by column. The ciphertext is taken from the state matrix after the last round column by column.

All steps of the round function *SubBytes*, *ShiftRows*, *MixColumns*, *AddKey* are invertible. Therefore, decryption with *Rijndeal* means to apply the inverse functions of *SubBytes*, *ShiftRows*, *MixColumns* in the reverse order.

2.3 The Round function

In this sub section we describe each steps (*SubBytes*, *ShiftRows*, *MixColumns*, *AddKey*) of the round function. The *Rijndeal* algorithm and its steps are byte-oriented.

They operate on the bytes of the state matrix. In *Rijndeal*, bytes are considered as elements of the finite field $GF(2^8)$ with 2^8 elements and $GF(2^8)$ is constructed as an extension of the field $GF(2)$ with 2 elements by using the irreducible polynomial $x^8+x^4+x^3+x+1$. Then adding and multiplying bytes means to add and multiply them as elements of the quotient field $GF(2)[x]/\langle Q(x)\rangle \cong GF(2^8)$.

2.3.1 The SubBytes

SubBytes[11] is the only non-linear transformation of *Rijndeal*. It substitutes the bytes of the state matrix byte by byte, by applying the function S_{RD} (Rijmen and Deamen's S-box) to each entries of the state matrix. The function S_{RD} does not depend on the key. S_{RD} is also called S-box. The same S-box is used for all byte positions. The S-box S_{RD} is composed of two maps f and g. That is:

$$S_{RD}(x) = (g \circ f)(x) = g(f(x)), (for\ all\ x \in GF(2_8)).$$

Both maps f and g, have a simple algebraic description. In order to understand f, we consider a byte x as an element of the finite field $GF(2^8)$. Then simply maps x to its multiplicative inverse x^{-1}. That is

$$\begin{aligned} f\colon GF(2^8) \to GF(2^8), f(x) &= x^{-1},\ if\ x \neq 0\ and \\ &= 0\ if\ x = 0. \end{aligned}$$

To understand g, we consider a byte x as vector of 8-bits or as a vector of length 8 over the field $GF(2)$. Then g is the $GF(2)$ affine map. That is:

$$g : GF(2^8) \to GF(2^8), x \to Ax + b$$

composed of a linear map $x \to Ax$ and a translation with vector b. The matrix A of the linear map and b are given by :

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \text{ and } b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

The S-box S_{RD} operates on each of the state bytes of the state matrix independently. For a block length of 128-bits, we have:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \begin{pmatrix} S_{RD}(a_{00}) & S_{RD}(a_{01}) & S_{RD}(a_{02}) & S_{RD}(a_{03}) \\ S_{RD}(a_{10}) & S_{RD}(a_{11}) & S_{RD}(a_{12}) & S_{RD}(a_{13}) \\ S_{RD}(a_{20}) & S_{RD}(a_{21}) & S_{RD}(a_{22}) & S_{RD}(a_{23}) \\ S_{RD}(a_{30}) & S_{RD}(a_{31}) & S_{RD}(a_{32}) & S_{RD}(a_{33}) \end{pmatrix}$$

2.3.2 The InvSubBytes

Definition 2.3.1. *[11]InvSubBytes is the inverse of the byte substitution SubByte.*

Which the inverse S-box S_{RD} is applied to each byte of the state. This obtained by applying the inverse of the affine transformation followed by taking the inverse in $GF(2^8)$. That is

$$S_{RD}^{-1}(x) = (f^{-1} \circ g^{-1})(x) = f^{-1}(g^{-1}(x)).$$

For the sake of simplicity, the inverse S-box S_{RD}^{-1} is presented as a look up table and the representation of x is in hexadecimal.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	54	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

2.3.3 ShiftRows

The *ShiftRows*[11] step transformation performs a cyclic shift to the state matrix. The offsets are different for each row and depends on the block length N_b.

N_b	1^{st}row	2^{nd}row	3^{rd}row	4^{th}row
4	0	1	2	3
5	0	1	2	3
6	0	1	2	3
7	0	1	2	4
8	0	1	3	4

for a block length of 128-bits (N_b) as in *AES*, *ShiftRows* is the map

Example 2.3.2. *Consider the case state matrix of AES-128, then*

$$ShiftRows: \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \rightarrow \begin{pmatrix} a & b & c & d \\ f & g & h & e \\ k & l & i & j \\ p & m & n & o \end{pmatrix}$$

2.3.4 InvShiftRows

Definition 2.3.3. *[11] InvShiftRows is the inverse of the ShiftRows function. The inverse operation is obtained by cyclic right shift with the same offsets that of ShitRows.*

Example 2.3.4. *Consider the case for state matrix of AES-128, then*

$$InvShitRows: \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \rightarrow \begin{pmatrix} a & b & c & d \\ h & e & f & g \\ i & j & k & l \\ p & o & n & m \end{pmatrix}$$

2.3.5 The MixColumn

The *MixColumn* [11] transformation operates on each column of the state matrix independently. We consider a column a=(a_0, a_1, a_2, a_3) as a polynomial $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$ of degree $\leq$3 with coefficients in $GF(2^8)$. Then the matrix transforms a column by multiplying it with the fixed polynomial $C(x) = 03x^3 + 01x^2 + 01x + 02$ and taking the residue of the product modulo $x^4 + 1$. That is

$$a(x) \rightarrow a(x)C(x)mod(x^4 + 1).$$

the coefficients of c are elements of $GF(2^8)$. Hence they are represented as bytes and a byte is given by two hexadecimal digits as usual.

The transformation of the *MixColumns*, multiplying $C(x)$ and taking the residue modulo x^4+1 are $GF(2^8)$ linear maps. Hence *MixColumns* is a linear map o vectors

of length 4 over $GF(2^8)$. It is given by the following 4×4 matrix over $GF(2^8$.

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

MixColumns transforms each column of the state matrix independently.

Example 2.3.5.

$$MixColumn : \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \rightarrow \begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{pmatrix}$$

where

$$\begin{pmatrix} b_{0j} \\ b_{1j} \\ b_{2j} \\ b_{3j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} a_{0j} \\ a_{1j} \\ a_{2j} \\ a_{3j} \end{pmatrix} j = 0, 1, 2, 3.$$

2.3.6 InvMixColumns

Definition 2.3.6. *[11]InvMixColumn is the inverse of the MixColumns transformation.*

InvMixColumns operates on the state matrix column by column treating each column as a four term polynomial, as it is mentioned in the previous subsection. Like the *MixColumns* step the *InvMixColumns* is also a linear map of vectors of length 4 over $GF(2^8)$. It is given by the following 4×4 matrix over $GF(2^8)$.

$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}$$

2.3.7 AddKey

AddKey[11] is the only operation in *Rijndeal* that depends on the secret *key K*. The *AddKey* operation adds a round key to the intermediate result state. The round keys are derived from the secret *key K* by applying the *key schedule algorithm*. Round keys are bit strings and, as the intermediate result state, they have block length, that is a sequence of N_b words. AddKey simply bitwise XORs the state with the round key to get the new value of state. That is:

$$AddKey : (state, round\ key) \rightarrow state \oplus roundkey.$$

Since we arrange state as a matrix, a round key is also represented as a round key matrix of bytes with 4-rows and N_b-columns. Each of the N_b-columns of the round key yields a column. And the new state matrix is obtained after the state matrix and the round key matrix are bitwise XORed by *AddKey*. XORing of two bytes means to add two elements of the field $GF(2^8)$.It is obvious that AddKey is invertible. AddKey is the inverse of itself.

2.4 The Key Schedule

The secret *key K* of N_k 4-byte words. The *Rijndeal* algorithm needs a round key for each round and one round key for the initial key addition. This means *Rijndeal* algorithm should generate N_r+1 round keys (N_r = number of rounds). A round key consists of N_b words. If we concatenate all the round keys, we get a string of $N_b(N_r+1)$ words. We call this string the *Expanded key*.

The *Expanded key* is derived from the secret *key K* by the key expansion procedure,

which we describe next. The round keys

$$Key_0, Key_1, Key_2, ..., Key_{N_r}$$

are then selected from the expanded key (ExpKey) Key_0 consists of the first N_b words of $ExpKey$, Key_1 consists of the next N_b words of ExpKey and so on.

In order to understand the key expansion procedure. We use functions f_j for multiples of j of N_k and a function g. All these functions map words (x_0, x_1, x_2, x_3) which each consists of 4-bytes of word.

g simply applies the S-box S_{RD} (as it is defined in the section 2.3.1.) to each byte:

$$g : (x_0, x_1, x_2, x_3) \rightarrow (S_{RD}(x_0), S_{RD}(x_1), S_{RD}(x_2), S_{RD}(x_3)).$$

If j is multiple of N_k, that is $j \equiv mod N_k$, we define f_j by:

$$f_j : (x_0, x_1, x_2, x_3) \rightarrow (S_{RD}(x_1) \oplus RC[j/N_k], S_{RD}(x_2), S_{RD}(x_3), S_{RD}(x_0)).$$

In other words f is composed of two functions *RotWod* and *SubWord* followed by XORing with *Round Constant RC*. That is:

$$(f_j(x_0, x_1, x_2, x_3) = SubWord(RotWord(x_0, x_1, x_2, x_3)) \oplus RC(j/N_k).$$

Where, *SubWord* = S-box S_{RD}, *RotWord* = one-offset left shift ,RotWord(f_j(x_0, x_1, x_2, x_3) = (f_j(x_1, x_2, x_3, x_0), Round Constant RC[i] $\in GF(2^8)$, defined by x^{i-1} $mod\, Q(x) = x^8 + x^4 + x^3 + x + 1$.

ExpKey[j], $0 \leq j \leq N_b(N_r+1)$ the words of the expanded key. The first N_k words are initialized with the secret *key K*. The following words are computed recursively. *ExpKey*[j] depends on *ExpKey*[j-N_k] and on ExpKey[j-1].

Depending on the key length N_k, there are two versions of the key expansion procedure, one for $N_k \leq 6$, the other for N_k ¿ 6. For $N_k \leq 6$ we have:

$$\begin{aligned} ExpKey[j] &= ExpKey[j - N_k] \oplus f_j(ExpKey[j-1]) if\ j\ \equiv mod N_k,\ or \\ &= ExpKey[j - N_k] \oplus ExpKey[j-1] \oplus if\ j\ ! \equiv mod N_k. \end{aligned}$$

If $N_k > 6$, we have:

$$\begin{aligned} ExpKey[j] &= ExpKey[j - N_k] \oplus f_j(ExpKey[j-1])\ if\ j \equiv mod N_k\ or \\ &= ExpKey[j - N_k] \oplus g(ExpKey[j-1]) if\ j\ ! \equiv mod N_k\ and\ j \equiv 4 mod N_k\ or \\ &= ExpKey[j - N_k] \oplus ExpKey[j-1]\ else. \end{aligned}$$

2.4.1 Key Expansion Algorithm

Key Expansion(byte Key[$4N_k$], word ExpKey[$N_b(N_r+1)$], N_k)
begin
word = *temp*
i=0
while(i$\leq N_k$)
ExpKey[i]=word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
i=i+1
end while
i=N_k
while(i > $N_b(N_r+1)$)
temp=ExpKey[j-1]
if(i $mod N_k$=0)
temp=SubWord(RotWord(temp))$\equiv$ RC[i/N_k]
else if ($N_k > 6$ and i $mod N_k$=4)
temp=SubWord(temp)
end if
ExpKey[i]-ExpKey[i-N_k]$\oplus$temp
i=i+1
end while
end

Example 2.4.1. *: Let N_k=4 and key_0, key_1, key_2, ..., key_{15} be the secret key. Then, according to the algorithm*

$$\begin{aligned} ExpKey[0] &= (key_0, key_1, key_2, key_3) \\ ExpKey[1] &= (key_4, key_5, key_6, key_7) \\ ExpKey[2] &= (key_8, key_9, key_{10}, key_{11}) \\ ExpKey[3] &= (key_{12}, key_{13}, key_{14}, key_{15}) \end{aligned}$$

i=4
temp=ExpKey[i-1]=ExpKey[3]=(key_{12}, key_{13}, key_{14}, key_{15})

$i\ mod N_k\ =\ 4\ mod 4=0$
$\Rightarrow\ temp=SubWord(RotWord(key_{12},\ key_{13},\ key_{14},\ key_{15}))\oplus RC[i/N_k]$
$=SubWord(key_{13},\ key_{14},\ key_{15},\ key_{12})\oplus Rc[i/N_k]$
$=(S_{RD}(key_{13}),\ S_{RD}(key_{14}),S_{RD}(key_{15}),S_{RD}(key_{12}))\oplus\ (x^{i-1} mod Q(x)= x^8\ +\ x^4\ +\ x^3\ +\ x\ +\ 1,0,0,0)$

2.5 Equivalent Inverse Cipher

In the inverse cipher the key schedule for encryption and decryption remains the same. However several properties of the *AES* allow for an Equivalent Inverse Cipher that has the same sequence of transformation as the cipher (with transformation replaced by their inverse). The two properties that allow for this Equivalent Inverse Cipher are as follows:

1.The SubBytes and ShiftRows transformations commute:

that is (SunByets$\circ$ShiftRows)(x)=(ShiftRows$\circ$SubBytes)(x) and the same is true for *InvSubBytes* and *InveShiftRows*.

2.The column mixing operation *MixColumns* and *InvMixColumns* are linear with respect to the column input, which means:

$InvMixColumns(State) \oplus roundkey)$
$= InvMixColumns(state) \oplus InvMixColumns(roundkey)$

Algorithm

InitState(Cipher text, state)

AddKey(state,Key_{N_r})

while (i= N_r-1 step -1 down to 1 i > 1) do

INSubBytes(state)

InvShiftRows(state)

InvMixColumns(state)

AddKey(state,key_i)

end while

InvSubBytes(state)

InvShiftRows(state)

AddKey(state,key_1)

return state.

Chapter 3

A Polynomial Representation of AES-128 Cipher

We have seen a detail description of Rijndeal algorithm. Now, we restrict our selves to AES-128. That is Rijndeal with block and key size of 128-bits.

So, by the definition given earlier for Block Cipher, definition of AES-128 is given by:

Definition 3.0.1. *AES-128 is a block cipher with* $M = C = K = \{0,1\}^{128}$*;*

$$AES - 128 : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}.$$

Notation: We will deviate form the standard representation; which uses a matrix for the internal state and round keys to a column vector. the elements in the column vector are identified with the elements of the matrix in a column-wise fashion by the map φ. That is

$$\varphi : GF(2^8) \to GF(2^8), \begin{pmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{pmatrix} \to \begin{pmatrix} s_{0,0}, & s_{1,0}, & s_{2,0}, & . & . & . & s_{2,3}, & s_{3,3}, \end{pmatrix}$$

We also define a 16×16 matrix P to be the permutation matrix that achieves the exchange of elements in the column vector that is equivalent to transposing the state matrix.

$$P = \begin{pmatrix} 01 & 00 & 00 & 00 & 00 & . & . & . & 00 \\ 00 & 00 & 00 & 00 & 01 & . & . & . & 00 \\ . & & & & & . & . & . & 00 \\ . & & & & & . & . & . & 00 \\ . & & & & & . & . & . & 00 \\ 00 & 00 & 00 & 00 & 00 & . & . & . & 01 \end{pmatrix} 16 \times 16$$

the entries are in hexadecimal.

The above notation allows us to express the diffusion performed by MixColuns and SiftRows operations as a single matrix multiplication. Let $x_{i,j}$ denote the variable referring to the i^{th} component of the state vector after the j^{th} round execution. According to this definition the variable $x_{i,0}$ are called the plaintext variables, corresponding $x_{i,10}$ are ciphertext variables. All other variables $x_{i,j}$ are called intermediate state variables. Whereas $k_{i,j}$ are called key variables. We will also refere to $K_{i,0}$ as cipher key variable (secrete key variable).

The field $\mathbf{F}$ is the finite fieled $GF(2^8)$ as defined for Rijndeal. The polynomial ring is defined as: $\mathbf{R}$= $\mathbf{F}[x_{i,j}$,$k_{i,j} : 0 \leq i \leq 15, 0 \leq j \leq 10]$

3.1 The S-box

As we have seen in the Preliminaries section, interpolating polynomial requires a set of n-data (n-points on a plane), (x_k,y_k), $k = 1, ..., n$ with x_k's are distinct. In Rinjndeal both the SubBytes and InvSubBytes are invertible (in other words one to one and on to).

Now, let consider the input and output of these functions as n-points (data), 256 points to be exact to construct our interpolating polynomial to represent the SubBytes and InvSubBytes functions. Since both SubBytes and InvSubBytes are one to one we are guaranteed that the inputs are distinct.

As a result we can construct the interpolation polynomials representing the SubBytes and InvSubBYtes functions. The S-box can be interpolated as a spare polynomial over $GF(2^8)$.

$$\sigma : GF(2^8) \rightarrow GF(2^8)$$

$$x \rightarrow 05x^{254} + 09x^{253} + f9x^{251} + 25x^{247} + f4x^{239} + b5x^{223} + b9x^{191} + 8fx^{127} + 63$$

While the interpolation polynomial of the inverse S-box (SubBytes operation)

$$\sigma^{-1} : GF(2^8) \rightarrow GF(2^8), x \rightarrow \sum_{i=0}^{254} c_i x_i$$

is dense.

3.2 The Linear Transformation

The linear transformation of AES consists of the two operations, ShiftRows and the MixColumns. Now, we will construct a 16X16 matrix **D**. Such that, multiplication of the state vector with this matrix **D**, will be the same as performing ShiftRows and MixColumns on the state matrix.

In order to make the operation easier we use the transposition matrix P defined earlier at the start of each round. At the end we also multiply with the same matrix P to undo the initial transposition.

3.2.1 The ShiftRows

A matrix that left shifts the element of 1×4 row vector cyclically by an offset t is of the following form:

$$D_{SRt} = (\triangle_{i,(j-t)mod4}) \in GF(2^8)^{4X4}.$$

Where $\triangle_{i,j}$ is the Kronecker delta.

Example 3.2.1. *D_{SRt} for t=2 that is offset of 2 is*

$$D_{SR2} = (\triangle_{i,(j-2)mod4}) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

The ShiftRows operation is equivalent to multiplying the state vector by the matrix D_{SR} :

$$D_{SR} : \begin{pmatrix} D_{SR1} & 0 & 0 & 0 \\ 0 & D_{SR1} & 0 & 0 \\ 0 & 0 & D_{SR2} & 0 \\ 0 & 0 & 0 & D_{SR3} \end{pmatrix}$$

3.2.2 The InvShiftRows

The InvShiftRows is an operation just like ShiftRows except a left cyclic shift in ShiftRows becomes a right cyclic shift in InvShiftRows with the same number of offset for each corresponding row.

A matrix that shifts the elements of 1X4 row vector cyclically by an offset t is of the following form:

$$D_{InSRt} = (\triangle_{(i-t)mod4,j}) \in GF(2^8)^{4X4}$$

Where $\triangle_{i,j}$ is the Kronecker delta.

The ShiftRows operation is equivalent to multiplying the state vector by the matrix D_{InSR} :

$$D_{InSR} : \begin{pmatrix} D_{InSR1} & 0 & 0 & 0 \\ 0 & D_{InSR1} & 0 & 0 \\ 0 & 0 & D_{InSR2} & 0 \\ 0 & 0 & 0 & D_{InSR3} \end{pmatrix}$$

3.2.3 The MixColumns

The MixColumns operation is applied to each row of the internal state we use the matrix D_{MC} to transform the column vector equivalently.

$$D_{MC} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \otimes I_4 \in GF(2^8)^{4X4}.$$

Where $\otimes$ is the tensor (Kronecker) product of two matrices. As a result

$$D_{MC} = \begin{pmatrix} 02[I_4] & 03[I_4] & 01[I_4] & 01[I_4] \\ 01[I_4] & 02[I_4] & 03[I_4] & 01[I_4] \\ 01[I_4] & 01[I_4] & 02[I_4] & 03[I_4] \\ 03[I_4] & 01[I_4] & 01[I_4] & 02[I_4] \end{pmatrix}$$

3.2.4 The InvMixColumns

The InvMixColumns operation is applied to each row of the internal state. We use the matrix D_{InMC} to transform the column vector equivalently.

$$D_{InMC} = \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 0e & 09 \end{pmatrix} \otimes I_4 \in GF(2^8)^{4X4}.$$

Where $\otimes$ is the tensor (Kronecker) product of two matrices. As a result

$$D_{MC} = \begin{pmatrix} 0e[I_4] & 0b[I_4] & 0d[I_4] & 09[I_4] \\ 09[I_4] & 0e[I_4] & 0b[I_4] & 0d[I_4] \\ 0d[I_4] & 09[I_4] & 0e[I_4] & 0b[I_4] \\ 0b[I_4] & 0d[I_4] & 09[I_4] & 0e[I_4] \end{pmatrix}$$

Concatenation of the operations ShiftRows and MixColumns is Achieved by multiplying the state vector by the matrix D,where D is given by:

$$D = P \circ D_{MC} \circ D_{SR} \circ P$$

The last round of AES is missing the MixColumns transformation, it will be described by the matrix $D^{'}$,

$$D^{'} = P \circ D_{SR} \circ P$$

Now, observe the following, for rounds $1 \leq j \leq 9$ of the cipher (AES-128),

$$D\begin{pmatrix} \sigma(x_{0,(j-1)}+K_{0,(j-1)}) \\ \sigma(x_{1,(j-1)}+K_{1,(j-1)}) \\ \sigma(x_{2,(j-1)}+K_{2,(j-1)}) \\ \sigma(x_{3,(j-1)}+K_{3,(j-1)}) \\ \sigma(x_{4,(j-1)}+K_{4,(j-1)}) \\ . \\ . \\ . \\ \sigma(x_{15,(j-1)}+K_{15,(j-1)}) \end{pmatrix} = \begin{pmatrix} x_{0,j} \\ x_{1,j} \\ x_{2,j} \\ x_{3,j} \\ x_{4,j} \\ . \\ . \\ . \\ x_{15,j} \end{pmatrix}$$

This enable us to obtain vectorial representation of a system of 16-polynomial equations that holds for rounds $1 \leq j \leq 9$ of the cipher.

$$\begin{pmatrix} \sigma(x_{0,(j-1)}+K_{0,(j-1)}) \\ \sigma(x_{1,(j-1)}+K_{1,(j-1)}) \\ \sigma(x_{2,(j-1)}+K_{2,(j-1)}) \\ \sigma(x_{3,(j-1)}+K_{3,(j-1)}) \\ \sigma(x_{4,(j-1)}+K_{4,(j-1)}) \\ . \\ . \\ . \\ \sigma(x_{15,(j-1)}+K_{15,(j-1)}) \end{pmatrix} + D^{-1}\begin{pmatrix} x_{0,j} \\ x_{1,j} \\ x_{2,j} \\ x_{3,j} \\ x_{4,j} \\ . \\ . \\ . \\ x_{15,j} \end{pmatrix} = 0$$

For the last round we need to take the simplified layer and the final key addition in to account.

$$\begin{pmatrix} \sigma(x_{0,9}+K_{0,9}) \\ \sigma(x_{1,9}+K_{1,9}) \\ \sigma(x_{2,9}+K_{2,9}) \\ \sigma(x_{3,9}+K_{3,9}) \\ \sigma(x_{4,9}+K_{4,9}) \\ . \\ . \\ . \\ \sigma(x_{15,9}+K_{15,9}) \end{pmatrix} + D'^{-1} \begin{pmatrix} x_{0,j}+K_{0,10} \\ x_{1,j}+K_{1,10} \\ x_{2,j}+K_{2,10} \\ x_{3,j}+K_{3,10} \\ x_{4,j}+K_{4,10} \\ x_{5,j}+K_{5,10} \\ . \\ . \\ . \\ x_{15,j}+K_{15,10} \end{pmatrix} = 0$$

Where $D^{-1} = P \circ D_{InSR} \circ D_{InMC} \circ P$ *and* $D'^{-1} = P \circ D_{InSR} \circ P$

3.3 The Key Schedule

In order to came up with system of polynomial equations for key schedule, we need to set up the key schedule in a slightly different way. Usually, the key scheduling express the elements of the round sub-key of round $1 \leq j \leq 10$ as a vector of polynomial in the key variable as follows:

$$\begin{pmatrix} K_{0,j} \\ K_{1,j} \\ K_{2,j} \\ K_{3,j} \\ K_{4,j} \\ . \\ . \\ . \\ K_{15,j} \end{pmatrix} = \begin{pmatrix} K_{0,j-1} \\ K_{1,j-1} \\ K_{2,j-1} \\ K_{3,j-1} \\ K_{4,j-1} \\ . \\ . \\ . \\ K_{15,j-1} \end{pmatrix} + \begin{pmatrix} \sigma(K_{15,j-1})+\gamma_{j-1} \\ \sigma(K_{12,j-1}) \\ \sigma(K_{13,j-1}) \\ \sigma(K_{14,j-1}) \\ K_{0,j} \\ . \\ . \\ . \\ K_{11,j} \end{pmatrix}$$

Where the γ_0 , . . ., γ_9 are the round constants. From the above equation we can have the following system of polynomial equations.

$$\begin{pmatrix} \sigma^{-1}(K_{0,j} + K_{0,j-1}\gamma_{j-1}) \\ \sigma^{-1}(K_{1,j} + K_{1,j-1}) \\ \sigma^{-1}(K_{2,j} + K_{2,j-1}) \\ \sigma^{-1}(K_{3,j} + K_{3,j-1}) \\ K_{4,j} + K_{4,j-1} \\ . \\ . \\ . \\ K_{15,j} + K_{15,j-1} \end{pmatrix} + \begin{pmatrix} K_{15,j-1} \\ K_{12,j-1} \\ K_{13,j-1} \\ K_{14,j-1} \\ K_{0,j-1} \\ . \\ . \\ . \\ K_{11,j-1} \end{pmatrix} = 0$$

Now, observe that we end up with, 200 polynomials of degree 254. The 160 polynomials of degree 254 is obtained from each round representation. The rest 40 from the key scheduling. That is the first four polynomials of the 10 round keys.

We also have linear polynomials, 120 in number. Which these linear polynomials obtained from the key scheduling.

Chapter 4

Constructing Groebner Basis

In this chapter we will explain how to construct a degree lexicographical Groebner basis describing the AES key recovery problem. But, first we give an introduction to Groebner basis, which we thought is enough for this thesis.

In the introduction section we try to explain Groebner basis of an ideal of polynomial ring is a standard representation, which has several useful properties, such as *Ideal membership*, *Solving system of polynomial equations*.

Let F be a field.

* *The ideal membership*[6]: Given $f \in K[x_1, ..., x_n]$ and an ideal $I =< f_1, ..., f_s >$, determine $f \in I$. If $G = f_1, ..., f_s$ is a Groebner basis, we say $f \in I$ if and only if the remainder of the division of f by G is zero.

* *Solving system of Polynomial equations*[6]: Find all common solutions in F^n of a System of polynomial equations:

$$\begin{aligned} f_1(x_1, ..., x_n) &= 0 \\ &. \\ &. \\ &. \\ f_s(x_1, ..., x_n) &= 0 \end{aligned}$$

4.1 Monomial Ordering

Groebner basis of an ideal I of $F[x_1, ..., x_n]$ is not unique. It depends on the term (monomial) ordering we are using.

Definition 4.1.1. *[6] A monomial (term) in $x_1, ..., x_n$ is a product of the form $x_1^{\alpha_1}, ..., x_n^{\alpha_n}$, where all of the exponents $\alpha_1, ..., \alpha_n$ are non-negative integers, the total degree of the monomial is the sum $\alpha_1 + ... + \alpha_n$.*
We can simplify the notion for the monomials as follows; let $\alpha = (\alpha_1, ..., \alpha_n)$ be an n-tuple non-negative integers. Then we set $X^\alpha = x^\alpha 1_1, ..., x^\alpha n_n$. When $\alpha = (0, ..., 0)$ then $X^\alpha = 1$.

Definition 4.1.2. *[6] A polynomial f in $x_1, ..., x_n$ with coefficients in F(field) is a finite linear combination (with coefficients in F) of the monomials, we will write a polynomial f in the form*

$$f = \sum_\alpha a_\alpha X^\alpha \ , \ a_\alpha \in F.$$

Definition 4.1.3. *[6] A subset $I \subset F[x_1, ..., x_n]$ is an ideal if it satisfies*
a.$0 \in I$
b.If $f, g \in I, then f + g \in I$
c.If $f \in I and h \in F[x_1, ..., x_n]$, the $fh \in I$

Note: For the rest of the paper R= $F[x_1, ..., x_n]$ and $T(R)$= the set of all terms.

Definition 4.1.4. *[6] Let $f_1, ..., f_s$ be polynomial in $F[x_1, ..., x_n]$. Then we set $< f_1, ..., f_s >= \sum h_i f_i : h_1, ..., h_s \in R.$ $< f_1, ..., f_s >$ is an ideal. See [6].*

Definition 4.1.5. *[6] (Term(monomial) order)[6] A term order $\leq$ is a linear (total) order on the set of terms T(R) such that*
a. $1 \leq t$ for all terms $t \in T(R)$
b. for all $s, t_1, t_2 \in T(R)$ whenever $t_1 \leq t_2$ then $st_1 \leq st_2$

Now, we will introduce useful and widely used term orders. First, we define two technicalities.

* For a term $t = x_1^{e1}, ..., x_n^{en} \in T(R)$ we define the exponent vector of t to be $e(t) = (e_1, ..., e_n) \in N_0^n$.

* Total degree of the term t then is $deg(t) = \sum_{i=1}^{n} e_i$.

Definition 4.1.6. *[6](Lexicographical). For terms s, t we define $s <_{lex} t$ if and only if the vector difference $e(s) - e(t) \in N_0^n$, then the left most non-zero entry is positive. We will write $s >_{lex} t$.*

Example 4.1.7. *[6]$e(s) = (3, 2, 4) \geq_{lex} e(t) = (3, 2, 1)$, where $s = x^3y^2z^4, t = x^3y^2z^1 \in R[x, y, z]$ because the first two terms of $e(s)$ and $e(t)$ are equal but the third term of $e(t)$, which is $1 \leq 4$, the third term of $e(s)$.*

Definition 4.1.8. *[6](degree lexicographical). For terms we define $s <_{dlex} t$ if and only if either $deg(s) < deg(t)$ or if $deg(s) = deg(t)$, then $s <_{lex} t$.*

Example 4.1.9. *$e(s) = (1, 2, 4) >_{dlex} e(t) = (1, 1, 5)$, where $s = xy^2z^4, t = xyz^5 \in R[x, y, z]$. Since $deg(s) = deg(t)$ and $(1, 2, 4) >_{dlex} (1, 1, 5)$.*

Definition 4.1.10. *[6](degree reverse lexicographic). For terms we define $s <_{drevlex} t$ if and only if either $deg(s) < deg(t)$ or if $deg(s) = deg(t)$, then vector difference $e(s) - e(t) \in N_0^n$, the right most non-zero entry is negative. We will write $s >_{drevlex} t$.*

Example 4.1.11. *$e(s) = (1, 5, 2) >_{drevlex} e(t) = (4, 1, 3)$, where $s = xy^2z^4, t = xyz^5 \in R[x, y, z]$. Since $deg(s) = deg(t)$ and $e(s) = (1, 5, 2) - e(t) = (4, 1, 3) = (-3, 4, -1)$*

And, now we will explain an elimination order.

Definition 4.1.12. *Fix an integer $1 \leq k \leq n$. We say that the monomial order $\geq$ on $F[x_1, ..., x_n]$ is of the $k-$elimination type provide that any monomial involving one of $x_1, ..., x_k$ is greater than all monomials in $F[x_{k+1}, ..., x_n]$.*

Definition 4.1.13. *Fix an integer* $1 \leq k \leq n$ *and* α *and* $\beta \in \mathbb{N}^n$. *Then we define the order* $>_k$ *as follows:* $\alpha >_k \beta$ *iff*

$$\sum_{i=1}^{k} \alpha_i > \sum_{i=1}^{k} \beta_i, or \sum_{i=1}^{k} \alpha_i = \sum_{i=1}^{k} \beta_i \text{ and } \alpha >_{drevlex} \beta.$$

Example 4.1.14. *1.* $(1,2,0) >_2 (2,0,1)$, *since the sum of the first two entries of* $(1,2,0)$ *is geater than the sum of the first two entries of* $(2,0,1)$.
2. $(1,2,0) >_2 (2,0,1)$, *since the sum of the first three entries of* $(1,2,0)$ *and* $(2,0,1)$ *are equal and* $(1,2,0) >_{drevlex} (2,0,1)$.

Definition 4.1.15. *[6] Let* $f = \sum_\alpha a_\alpha X^\alpha$ *be a non-zero polynomial in* $F[x_1, ..., x_n]$ *and let* $\geq$ *be a term order.*
a. The multidegree of f is
$multideg(f) = max(\alpha \in \mathbb{N}_0^n : a_\alpha \neq 0)$ *(the max is taken w.r.t.* $\geq$*)*
b. The leading coefficient of f is
$LC(f) = a_{multideg(f)} \in F$
c. The leading monomial of f is
$LM(f) = x^{multideg(f)}$ *(with coefficient 1)*
d. The leading term of f is
$LT(f) = LC(f)LM(f)$.

Now, we will define Groebner basis. But after the definition has been given one can see that the formal definition of the Groebner basis does not give much insight about how to construct one.

Definition 4.1.16. *[6](Groebner basis): Let I be an ideal of R. A set of polynomial* $\{g_1, ..., g_m\} \subset I$ *is a Groebner basis if the following holds:*

$$< LT(g_1), ..., LT(g_m) > \quad = \quad < LT(P) : P \in I >.$$

Note: For all $f \in I$ division of f by $G =$ Groebner basis of I in any order is zero[6].

Before we try to give some example and construction of Groebner basis, let us introduce ourselves to some terms, S-polynomial and Buchberger's criterion.

Definition 4.1.17. *[6] Let* $f, g \in R$ *be non-zero polynomials.*
a. If $e(LM(f)) = (t_1, ..., t_n)$ *and* $e(LM(g)) = (s_1, ..., s_n)$, $LM(f)$ *and* $LM(g) \in T(R)$,

then let $(r_1, ..., r_n)$*, where* $r_i = max(t_i, s_i)$ *for each* i*. We call* $X^r = \prod_{i=1}^{n} x_i^{r_i}$ *the least common multiple of* $LM(f)$ *and* $LM(g)$*, write* $X^r = (LCM(LM(f), LM(g))$
b. The S-polynomial of f *and* g *is the combination:*
$S(f,g) = \frac{X^r}{LT(f)} * f - \frac{X^r}{LT(g)} * g$

Example 4.1.18. *Let* $f = x^3y^2 - x^2y^3 + x$ *and* $g = 3x^3y + y^2 \in R[x, y]$ *with dlex order. Then* $LCM(LM(f), LM(g)) = x^4y^2$

$$\begin{aligned} S(f,g) &= \frac{x^3y^2}{x^3y^2} * f - \frac{x^3y^2}{3x^3y} * g \\ &= f - \frac{1}{3} * y * g \\ &= -x^2y^3 + x^2 - \frac{1}{3} * y^3 \end{aligned}$$

An S-polynomial $S(f, g)$ is designed to produce cancellation of the leading terms.

4.2 Buchberger's Algorithm

Buchberger's Algorithm, is an algorithm used to construct a Groebner basis. It uses first Buchberger criterion [6] to avoid useless polynomial reduction.

Buchberger criterion

Let I be a polynomial ideal. Then a basis $G = \{g_1, ..., g_s\}$ is a Groebner basis for I if and only if all pairs $i \neq j$, the remainder on division of $S(g_i, g_j)$ by G (in some order) is zero.

Proof

($\Rightarrow$) suppose G is a Groebner basis.

Now, since $S(g_i, g_j) \in I$ and G is a Groebner basis by hypothesis, the remainder on division of $S(g_i, g_j)$ by G is zero.

($\Leftarrow$) Let $f \in I$. We have to show that if the s-polynomials all have zero remainder on division by G, the $LT(f) \in < LT(g_1), ..., LT(g_t) >$.

one can see, given $f\ inI =< g_1, ..., g_t >$, there are polynomials $h_i\ inF[x_1, ..., x_n]$ such that

$$f = \sum_{i=1}^{t} h_i g_i$$

This implies $multideg(f) \leq max(multideg(h_i g_i))$

Let $m(i) = multideg(h_i g_i)$,

$\delta = max\{m(1), ..., m(t)\}$.

Then we have $multideg(f) \leq \delta$.

Now consider all possible ways that f can be written in the form

$$f = \sum_{i=1}^{t} h_i g_i.$$

For each expression, we get possibly different δ. Since monomial order is well ordering, we can select an expression for f such that δ is minimal.

Observe the following : If $multideg(f) = multideg(h_i g_i)$ for some i, then $LT(f)$ is divisible by $LT(g_i)$. This will show that $LT(F) \in \ < LT(g_1), ..., LT(g_t) >$. This is what we need to prove.

Claim: $multideg(f) = \delta$

Suppose not, that is equality fails. But equality fails only when $multideg(f) < \delta$.

Now, lets rewrite f in the following form:

$$f = \sum_{m(i)=\delta} h_i g_i + \sum_{m(i)<\delta} h_i g_i = \sum_{m(i)=\delta} LT(h_i) g_i + \sum_{m(i)=\delta} (h_i - LT(h_i)) g_i + \sum_{m(i)<\delta} h_i g_i$$

Observe that multidegree of the second and third sum are $lessthan\delta$. Since $multideg(f) \leq \delta$ means that the first sum also has multidegree $lessthan\delta$.

Let $LT(h_i) = c_i x^{\alpha(i)}$. This implies

$\sum_{m(i)=\delta} LT(h_i)g_i = \sum_{m(i)=\delta} c_i x^{\alpha(i)} g_i$.

Then the sum can be written as a linear combination of the S-polynomials $S(x^{\alpha(j)}g_j, x^{\alpha(k)}g_k)$ (By Lemma 5 of Chapter 2 of [6]). However

$$S(x^{\alpha(j)}g_j, x^{\alpha(k)}g_k) = \frac{x^\delta}{x^{\alpha(j)}LT(g_j)} * x^{\alpha(j)}g_j - \frac{x^\delta}{x^{\alpha(k)}LT(g_k)} * x^{\alpha(k)}g_k = x^\delta - \gamma_{jk}S(g_j, g_k).$$

Where $x^{\gamma_{jk}} = LCM(LM(g_j), LM(g_k))$. Thus there are constants $C_{jk} \in F$ such that

$$\sum_{m(i)=\delta} h_i g_i = \sum c_{jk} x^{\delta-\gamma_{jk}} S(g_j, g_k)$$

By hypothesis we have

$$S(g_j, g_k) = \sum_{i=1}^{t} a_{ijk} g_i, a_{ijk} \in F[x_1, ..., x_t]$$

$\Rightarrow multideg(a_{ijk}g_i) \leq multideg(S(g_j, g_k))$ (by division algorithm [6])

$$\Rightarrow x^{\delta-\gamma_{jk}} S(g_j, g_k) = \sum_{i=1}^{t} b_{ijk} g_i$$

Where $b_{ijk} = x^{\delta-\gamma_{jk}} a_{ijk}$

$$\Rightarrow \sum LT(h_i g_i) = \sum_{j,k} C_{jk}(\sum b_{jk} g_{jk}) = \sum h_i' g_i$$

this implies $multideg(h_i' g_i) \leq \delta$.

Now we have a new a expression for f as a combination of $g_i's$, which contradicts the minimality of δ. This concludes our proof.

Buchberger Algorithm

Let $I =< f_1, ..., f_s >\neq 0$ be a polynomial ideal. Then a Groebner basis for I can be constructed in finite number of steps by the following algorithm:

Input: $F = (f_1, ..., f_s)$
Output: a Groebner basis $G = (g_1, ..., g_t)$ for I, with $F \subset G$
$G := F$
REPEAT
$G' = G$
For each pair $p, q, p \neq q$ in G' DO
$S := remainder of S(p, q) by G$
If $S \neq 0 Then G := G \cup S$
Until $G = G'$.

Example 4.2.1. *Let* $I =< f_1, f_2 >=< xz - y^2, x^3 - z^2 >\subseteq C[x, y, z]$, *and use dlex order. Let* $f = -4x^2y^2 + y^6 + 3z^5$. *We want to know if* $f \in I$.
Observe that the generating set given is not a Groebner basis of I *because* $< LT(I) >$ *also contain polynomial such that*

$$LT(S(f_1, f_2)) = LT(-x^2y^2 + z^2) = x^2y^2 that is not in the ideal$$

$$< LT(f_1), LT(f_2) >=< xz, x^3 > .$$

Hence we began by computing a Greobner basis for I.

Using the Buchberger algorithm we find a Groebner basis

$$G = \{f_1, f_2, f_3, f_4, f_5\} = \{xz - y^2, x^3 - z^2, x^2y^2 - z^3, xy^4 - z^4, y^6 - z^5\}$$

Now, we may test polynomial for membership in I. *For example, dividing* f *above by* G *we find*

$f = (-4x^2y^2 + y^6 + 3z^5) = f_1 + 0 * f_2 + 0 * f_3 + 0 * f_4 + (-3) * f_5 +)$
Since the remainder is zero, we have $f \in I$.

Example 4.2.2. *Consider the equations*

$$\begin{aligned} f_1 &= t^2 + x^2 + y^2 + Z^2 = 0 \\ f_2 &= t^2 + 2x^2 - xy - z^2 = 0 \\ f_3 &= t + y^3 - z^3 = 0 \end{aligned}$$

The Groebner basis for $I = \langle f_1, f_2, f_3 \rangle$ *with respect to the lex order with* $t > x > y > z$ *is*

$$g_1 = y_{12} - 4z_3y_9 + 5y_8 + 6z_2y_6 - 10z^3y^5 + 5y^4 - 4z^9y^3 - 12z^5y^3 + 5z^6y^2 + 13z^2y^2 + z^{12} + 6z^8 + 9z^4$$

$$g_2 = xz^6 + 3xz^2 - y^{11} + 4y^8z^3 - 5y^7 - 5y^5z^6 - 3y^5z^2 + 10y^4z^3 - 5y^3 + 2z^9y^2 + 6y^2z^5 - 3yz^5 - 7yz^2$$

$$g_3 = xy + y^6 - 2y^3z^3 + 2y^3z^3 + 2y^2 + z^6 + 3z^2$$

$$g_4 = t^2 + x^6 - 2y^3z^3 + y^2 + z^6 + 3z^2$$

$$g_5 = t + y^2 - z^3$$

And, the Groebner basis for I with respect to first elimination order with $t > x > y > z$ *is*

$$\begin{aligned} h_1 &= x^2 - xy - y^2 - z^2 - z \\ h_2 &= y^6 - 2y^3z^3 + z^6xy + 2y^2 + 2z^2 + z \\ h_3 &= t + y^3 - z^3 \end{aligned}$$

The next theorem follows almost instantaneously from Buchberger criterion and gives an important hint how a Groebner basis can be attained without knowing anything about polynomial reduction.

Theorem 4.2.3. *Let* G *be a set of polynomials and* $H = \{LT(f) : f \in G\}$. *If all elements in* H *are pairwise prime, then* G *is a Groebner basis.*

Proof:

Recall Buchberger Criterion. That is $I =< g_1, ..., g_s >$, then G is a Groebner basis iff the remainder of division of $S(g_i, g_j)$ by G is zero for all $i \neq j$.

Consider g_i, g_j for $i \neq j$, then

$S(g_i, g_j) = \dfrac{x^r}{LT(g_i)} * g_i - \dfrac{x^r}{LT(g_j)} * g_j$,$x^r = LCM(LM(g_i), LM(g_j))$.

Since, $LT(g_i)$ and $LT(g_j)$ are relatively prime (by hypothesis), we have $x^r = LCM(LM(g_i), LM(g_j)) = LM(g_i) * LM(g_j)$

$$\Rightarrow S(g_i, g_j) = \frac{LM(g_i) * LM(g_j)}{LT(g_i)} * g_i - \frac{LM(g_i) * LM(g_j)}{LT(g_j)} * g_j \ .$$
$$= \frac{1}{LC(g_i) * LM(g_i)} * g_i \frac{1}{LC(g_j) * LM(g_j)} * g_j$$

$\Rightarrow$ the remainder of division of $S(g_i, g_j)$ by G is zero.

Since $g_i and g_j$ were arbitrary in G, G is Groebner basis.

In order to solve system of polynomial equations, we generate an ideal with the polynomial, and find the set of common zeros, called *variety of the ideal*. While different basis may generate the same ideal, the variety of the ideal will always be the same.

Definition 4.2.4. *[6] Let F be a field, and let $f_1, ..., f_s$ be polynomials in $F[x_1, ..., x_n]$. Then we set $V(f_1, ..., f_s) = \{(a_1, ..., a_n \in F^n : f_i(a_1, ..., a_n) = 0 \forall 1 \leq i \leq n\}$ the affine variety defined by $f_1, ..., f_s$.*

Definition 4.2.5. *[6] A Groebner basis $G = \{g_1, ..., g_t\}$ is reduced if*

$$1. LC(g_i) = 1, for all g_i \in G i = 1, ..., t$$
$$2. For all\ g_i \in G, no\ monomial\ of\ g_i\ lies\ in \langle\ LT(G - \{g_i\})\rangle.$$

A zero dimensional ideal is an ideal that has a finite number of solutions over the closure of the field. It usually is advantageous to have this property for Groebner basis computations. By using the next Lemma we can determine whether an ideal is zero dimensional or not.

Lemma 4.2.6. *[1] Let I be a proper ideal of $F[x_1, ..., x_n]$. Then the following assertion are equivalent.*
a. $dim(I) = 0$
b. There exists a term order $\leq$ such that for each $1 \leq i \leq n$ there is $g_i \in I$ with $LT(g_i) = x_i^{vi}$ for some $0 \leq v_i \in N$.

Example 4.2.7. *Consider the ideal* $I = \langle xy - z, yz - x, zx - y \rangle$. *Choose lexicographic monomial ordering with* $x < y < z$, *we obtain a Groebner basis* $G = \{x^3 - x, yx^2 - y, y^2 - x^2, z - yx\}$.
Observe that, $\exists\ g_i \in G$ *suchthat* $LT(g_i) = a^{v(i)}, a \in \{x,\ y,\ z\}$.
that is

$$\begin{aligned} LT(x^3 - x) &= x^3 \\ LT(y^2 - x^2) &= y^2 \\ LT(z - yx) &= z \end{aligned}$$

Therefore according to the above Lemma I is zero dimensional ideal.

4.3 A Zero Dimensional Groebner basis for AES-128

So far we have seen, how to construct a Groebner basis . Now we give an example on how the construction of Groebner basis depends on the term order we are using.

Example 4.3.1. *Let* $I =< f_1 = x^2 + y^2 - z^2, f_2 = x^3 + y > \subseteq R[x, y]$. *We are going to find a Groebner basis for* I *both with respect to lex and dlex. But with respect to each term order the generating set is not a Groebner basis.*

$S(f_1, f_2) = xy^2 - xz^2 - y$ *w.r.t lex, but division of* $S(f_1, f_2) = xy^2 - xz^2 - y$ *by* G *is not equal to o, and*
$S(f_1, f_2) = xy^2 - xz^2 - y$ *w.r.t lex, but division of* $S(f_1, f_2) = xy^2 - xz^2 - y$ *by* G *is not equal to o.*
Now using computer algebra (Singular) we get the following two different Groebner basis for I *for each respective term order.*
That is, $(x^2 + y^2 - z^2, y^6 - 3y^4z^2 + 3y^2z^4 + y^2 - z^2, xz^5 + y^5 - 2y^3z^2 + yz^4 + y)$ *w.r.t. lex and* $(x^2 + y^2 + z^2, xy^2 - xz^2 - y, y^4 - 2y^2z^2 - z^4 + xy)$ *w.r.t. dlex.*

4.3.1 Choosing a Suitable Variable Order

Now lets consider the polynomial equation we have obtained in the 'Changing AES-128 to Polynomial Equation' section.

For rounds $1 \leq j \leq 9$ of the cipher.

$$\begin{pmatrix} \sigma(x_{0,(j-1)} + K_{0,(j-1)}) \\ \sigma(x_{1,(j-1)} + K_{1,(j-1)}) \\ \sigma(x_{2,(j-1)} + K_{2,(j-1)}) \\ \sigma(x_{3,(j-1)} + K_{3,(j-1)}) \\ \sigma(x_{4,(j-1)} + K_{4,(j-1)}) \\ . \\ . \\ . \\ \sigma(x_{15,(j-1)} + K_{15,(j-1)}) \end{pmatrix} + D^{-1} \begin{pmatrix} x_{0,j} \\ x_{1,j} \\ x_{2,j} \\ x_{3,j} \\ x_{4,j} \\ . \\ . \\ . \\ x_{15,j} \end{pmatrix} = 0$$

For the last round we need to take the simplified layer and the final key addition in to account.

$$\begin{pmatrix} \sigma(x_{0,9} + K_{0,9}) \\ \sigma(x_{1,9} + K_{1,9} \\ \sigma(x_{2,9} + K_{2,9}) \\ \sigma(x_{3,9} + K_{3,9}) \\ \sigma(x_{4,9} + K_{4,9}) \\ . \\ . \\ . \\ \sigma(x_{15,9} + K_{15,9}) \end{pmatrix} +^{'-1} \begin{pmatrix} x_{0,j} + K_{0,10} \\ x_{1,j} + K_{1,10} \\ x_{2,j} + K_{2,10} \\ x_{3,j} + K_{3,10} \\ x_{4,j} + K_{4,10} \\ . \\ . \\ . \\ x_{15,j} + K_{15,10} \end{pmatrix} = 0 \tag{4.3.1}$$

For the key schedule

$$\begin{pmatrix} \sigma^{-1}(K_{0,j} + K_{0,j-1}\gamma_{j-1}) \\ \sigma^{-1}(K_{1,j} + K_{1,j-1}) \\ \sigma^{-1}(K_{2,j} + K_{2,j-1}) \\ \sigma^{-1}(K_{3,j} + K_{3,j-1}) \\ . \\ . \\ . \\ K_{15,j} + K_{15,j-1} \end{pmatrix} + \begin{pmatrix} K_{15,j-1} \\ K_{12,j-1} \\ K_{13,j-1} \\ K_{14,j-1} \\ K_{0,j} \\ . \\ . \\ . \\ K_{11,j-1} \end{pmatrix} = 0 \tag{4.3.2}$$

The plaintext and ciphertext polynomials simply are the form

$x_{i,0} + C_i, C_i \in F, 0 \leq i \leq 15$

respectively

$x_{i,0} + C_i, C_i \in F, 0 \leq i \leq 15.$

where the P_i and C_i are the known plain text cipher text pairs we take in to consideration in the beginning.

Let Λ be the union of the left side of the above equations (4.3.1.) and (4.3.2.) for all rounds $i \leq j \leq 10$ as well as the plaintext and ciphertext polynomials. The 120 linear polynomials we have obtained become 152 including the above 32 linear polynomials involving the known plaintext/ciphertext pair.

By choosing degree lexicographical term order, either a term $x_{i,j}^{254}$ or a term $k_{i,j}^{254}$ occurs as a head term of each polynomial. We take note that none of the head terms is a power of a plaintext nor of a ciphertext variable.

We want to use Lemma 4.2.6. and Theorem 4.2.3. to obtain a zero dimensional Groebner basis for AES-128. In order to achieve our goal we need a fixed variable

order. The variable order choosen will influence whether the leading term is power of a key variable or of an intermediate state variable. Ordering the variables as follows makes all the head terms pairwise prime:

1. plaintext variables: $x_{0,0} < ... < x_{15,0}$
2. ciphertext variable: $x_{0,10} < ... < x_{15,10}$
3. key variables of all rounds in natural order: $k_{0,0} < k_{1,0} < ... < k_{15,10}$
4. intermediate state variables in their natural order.

The degree lexicographical term order with the above variable order will be in the following be refereed to as $<_\Lambda$.

By Theorem 4.2.3., the set of polynomials Λ is a Groebner basis to their term order. Moreover, using Lemma 4.2.6. we verify that ideal is zero dimensional.

Chapter 5

Cryptanalytic Implication of A-zero Dimensional Groebner basis of AES-128

In the previous section we have shown how to obtain a Zero Dimensional Groebner basis for AES-128. In this section we explain and discuss the cryptanalytic implication of this result which we investigate and discuss Groebner basis conversion, find an invariant under the elimination of the variables and explain the naive way of applying the ideal membership text, guessing parts of the round keys.

5.1 Groebner Basis Conversion

An obvious question is whether the Groebner basis we have computed in the previous section can be efficiently converted to a different, more suitable order, i.e a lexicographic order or elimination order.

Two algorithms and variations, FGLM algorithm[2] and the Groebner Walk[4] are known for performing Groebner basis conversion. While the FGLM algorithm only works for a zero-dimensional ideals, the Groebner Walk naturally also works for ideals of posetive dimension. Since we have established Λ, a zero dimensional, we are in

position to use FGLM.

An important characteristic of the ideal is the vector space dimension of the residue class ring obtained when factoring the polynomial ring R by the ideal I:

Definition 5.1.1. *[1] Let $R := F[x_1, ..., x_n]$. Then the F-space dimension of the ideal $I \subset R$ shall be denoted by dim(R/I).*

5.1.1 FGLM Algorithm [2]

The name $FGLM$ algorithm was developed by J.C. Fraugere, P.Gianni, D.Lazard and T.Mara. The drawback to this algorithm is that it only applies to zero-dimensional ideals.

The algorithm, first, of course one must have an initial Groebner basis, G, with respect to an initial monomial order. The algorithm then progress through three steps for each of the monomials in the ring $F[x_1, ..., x_n]$. At the beginning of each loop, there will be two sets G_{new}, which is initially empty but will become the new Groebner basis for the desired monomial order, and B, which is also initially empty but will grow to be the basis of the quotient ring $F[x_1, ..., x_n]/I$ as a $F - vectorspace$.

The $FGLM$ algorithm consists of three main parts

1.**Main Loop**: For this step the user will take the current input monomial X_α, initially 1, and find the remainder under division by G, denoted by $X^\alpha modG$. Recall that the G is the Groebner basis of the ideal with respect to the original monomial ordering. There are two possible cases for what will happen to the remainder. For the first, if the remainder under division G of X^α is linearly dependent on the remainder of other members of B, then we have a linear combination such that

$$X^\alpha modG - \sum c_j X^{\alpha(j)} modG = 0$$

where $X^{\alpha(j)} \in B \; and \; c_j \in F$. This implies that

$$g = X^{\alpha} - \sum c_j X^{\alpha(j)} \in I.$$

So we add g to the list G_{new} as the last element. Because we work through the various X^{α} in increasing order with respect to the new ordering, a polynomial g that added to G_{new} will always have X^{α} with coefficient of 1 as its leading term

In the second case, $X^{\alpha} mod G$ is linearly independent of the remainders of the items in B. In this event, X^{α}, is added to B.

If the first case applied and we added a polynomial to G_{new}, then G_{new} must be tested to see if it is the desired Groebner basis. To do this, we use the Termination test, the second part of the $FGLM$ algorithm.

2. **Termination Test**: In this event that a new polynomial, g, was added to G_{new}, the user must compute $LT(g)$. If the $LT(g)$ is power of x_i, where x_i is the greatest variable in the new monomial ordering, then the algorithm terminates. Otherwise proceed with to the third part of the algorithm, Next Monomial step.

3. **Next Monomial**: In this phase of the algorithm, simply replace the X^{α} that has just been processed with the next monomial with respect to the new order which is not divisible by any of the leading terms of the polynomials in G_{new}.

The user repeats the steps of this algorithm until the conditions are met for the Termination Test.

Notice that whenever a polynomial g is added to G_{new}, its leading term $LT(g) = X^{\alpha}$ with coefficient 1, hence each basis element must be monic. Also, because the leading term of each basis element is linearly independent of the leading terms of all other elements, the Groebner basis obtained from this algorithm must be reduced. The following example fully demonstrates the algorithm.

Example 5.1.2. *We will use the FGLM algorithm to find a lexicographic order (with $x > y > z$)Groebner basis for the ideal $I = \langle x^2+2y^2-y-2z, x^2-8y^2+10z-1, x^2-7yz\rangle$ from the degree reverse lexicographic Grobner basis, $G = \langle 980z^2 - 18y - 120z + 13, 35yz - 4y + 2z - 1, 10y^2 - y - 12z + 1, 5x^2 - 4y + 2z - 1\rangle$. We start with the least variable in the monomial order,z and consider it raised to the 0 degree. We then calculate the remainder of z^0 under division by G. We then continue increasing the degree of z and finding remainders under division by G until we find a remainder that is linearly dependent upon the other remainders. This linearly dependent remainder is subtracted from the dividend for which it corresponds and this polynomial is added to the set making up the Groebner basis.*

$$\begin{aligned} z^0 mod(G) &= 1 mod(G) = 1 \\ z mod(G) &= z \\ z^2 mod(G) &= \frac{9}{490}y + \frac{201}{980}z - \frac{13}{980} \\ z^3 mod(G) &= \frac{2817}{480200}y + \frac{26653}{960400}z - \frac{2109}{960400} \end{aligned}$$

Since $z^3 mod(G)$ is a linear combination of $z^0 mod(G), z^1 mod(G), z^2 mod(G)$,

$$g_1 = z^3 - \frac{313}{980}z^2 + \frac{37}{980}z + \frac{1}{490}$$

is the first polynomial added to the G_{new}.

Now we consider the next variable in the monomial order, y. We again take the remainder with respect to G.

$$y mod(G) = y$$

We find y itself can be expressed as a linear combination, namely

$$\begin{aligned} y &= \frac{490}{9}z^2 mod(G) - \frac{67}{6}z mod(G) + \frac{13}{18} \text{ and subsequently} \\ g_2 &= y - \frac{490}{9}z^2 + \frac{67}{6}z - \frac{13}{18} \end{aligned}$$

is added to G_{new}.

Lastly consider the greatest variable in the order, x.

$$\begin{aligned} x mod(G) &= x \\ x^2 mod(G) &= \frac{4}{5}y - \frac{2}{5}z + \frac{1}{5} \end{aligned}$$

Now $x^2 mod(G)$ can be expressed in relation to $ymod(G)$ and $zmod(G)$ and accordingly

$$g_3 = x^2 - \frac{392}{9}z^2 + \frac{28}{3}z - \frac{7}{9}$$

is the final polynomial added to G_{new}, leaving us with

$$G = \{g_1, g_2, g_3\}$$

Our desired lex Groebner basis.

Before we show that the FGLM algorithm is a valid method for changing the ordering of a Groeber basis. Now we state the following lemma:

Lemma 5.1.3. *[6](Dickson's Lemma) Given an infinite list $x^{\alpha(1)}, x^{\alpha(2)}, ...$ of monomials in $F[x_1, ..., x_n]$, there is an $N \in \mathbb{N}$ such that every $x^{\alpha(i)}$ is divisible by one of $x^{\alpha(2)}, x^{\alpha(2)}, ..., x^{\alpha(N)}$.*

Theorem 5.1.4. *[2] The algorithm described above terminates on every input Groebner basis, G, that generates a zero-dimensional ideal I, and correctly computes a lex Groebner basis, G_{new}, for I and the new monomial basis, B, for the quotient ring $F[x_1, ..., x_n]/I$.*

Proof

Observe that monomials added to the list B in strictly increasing new order. Similarly, if $G_{new} = \{g_1, ..., g_k\}$, then

$$LT(g_1) < ... < LT(g_k)$$

Also note that every time the Main Loop adds a new polynomial g_{k+1} to $G_{new} = \{g_1, ..., g_k\}$ to the leading term $LT(g_{k+1})$ is the input monomial in the Main Loop.

Since the input monomial are provided by Next Monomial procedure, the we have, for all $k, LT(g_{k+1})$ is divisible by none of $LT(g_1, ..., LT(g_k)$.

Now, we proof the termination of the algorithm.

Suppose not!

i.e. the algorithm does not terminate for some input G, then the Main Loop will be excuted infinite times.

$\rightarrow$ one of the two cases in the Main Loop would be chosen infinitely.

If the first alternative were chosen infinitely,

$\Rightarrow G_{new}$ would give an infinite list

$$LT(g_1), LT(g_2), LT(g_3), ...) \; of \; monomials.$$

But this is a contradiction with Dickson's Lemma ($\because$ there is no $k \; in \; \mathbb{N}$ such that $LT(g_{k+1})$ is divisible by none of $LT(g_1), ..., LT(g_k)$).

If the second case were chosen infinitely, then B would give infinitely many monomials $x^{\alpha(i)}$ whose remainder on division by G were linearly independent.

But, this contradicts, the assumption that I is a zero-dimensional.

$\therefore$ the algorithm always terminates if G generates a zero-dimensional ideal I.

Now, we will proof a consecutive lemmas and proposition which will help us estimate the time complexity of FGLM.

5.1.2 Complexity of FGLM

Definition 5.1.5. *[1] If $\leq$ is a term order on $T(R)$, then the set of reduced terms with respect to I and $\leq$ defined as $T(R)/LT(I)$ and denoted by $RT(I)$.*

Lemma 5.1.6. *[1] Let $\leq$ be a term order on $T(R)$, G a Groebner basis of I with*

respect to $\leq$. *Then*

$$\begin{aligned} RT(I) &= \{t \in T(R) : s \text{ does not divide } t \ \forall \ s \ \in LT(I)\} \\ &= \{t \in T(R) : s \text{ does not divide } t \ \forall \ s \ \in LT(G)\} \end{aligned}$$

Proof

Let $a \in RT(I) = T(R)/LT(I)$

$\Leftrightarrow a = t + LT(I) = t + LT(0)$

$\Leftrightarrow$ there is no $s \in LT(I)$ such that s divides t

$\Leftrightarrow a \in \{t \in T(R) :$ s does not divide $t \forall s \in LT(I)\}$

Now let $a \in \{t \in T(R) : s$ does not divide $t \ \forall \ s \ \in LT(I)\}$ there is no $s \in LT(I)$ such that s/a

$\Rightarrow$ there is no $p \in LT(G)$ such that p/a ($\because G \subset I$)

$\Rightarrow a \in \{t \in T(R) : s$ does not divide $t \ \forall s \in LT(G)\}$

$\Rightarrow \{t \in T(R) : s$ does not divide $t \ \forall s \in LT(I)\} \subset \{t \in T(R) : s$ doe not divide $t \ \forall s \in LT(G)\}$

The conversion

Let $b \in \{t \in T(R) : sdoe \text{ not divide } t \forall s \in LT(G)\}$

$\Rightarrow$ *there is no* $p \ \in LT(G)$ *suchthat* p/b

Claim: $b \in \{t \in T(R) : s$ does not divide $t \forall s \in LT(I)\}$

Suppose not!

$\Rightarrow \exists q \in LT(I) suchthat \ q/b$

Since $q \in LT(I)$ we have $q = rp$ for some $p \in LT(G)$ *and* $r \in LT(R)$ ($\because$ G is Groebner basis)

$\Rightarrow p/qandq/b$

$\Rightarrow p/b \ for \ p \in LT(G)$ which is a contradiction with our assumption

Proposition 5.1.7. *[1] Let B be basis of R/I. Then the map Q: RT(I)* $\rightarrow$ *B given by*

$t \to t + I$ *is bijective.*

Proof:

one to one

Let $t, r \in RT(I)$

$Q(t) = Q(r)$

$\Rightarrow t + I = r + I$

$\Rightarrow t - r \in I$

$\Rightarrow t - r$ is divisible by p, for some $p \in LT(G)$

But since all t, r, p are terms and we have

p does not divide $t \Rightarrow \ multideg(t) < multideg(p)$ and

p does not divide $r \Rightarrow \ multideg(r) < multideg(p)$ and

we also have $multide(t - r) \leq multideg(t) < multideg(p)$

$\Rightarrow p$ does not divide $(t - r)$ unless $(t - r) = 0$

$\Rightarrow t = r$

on-to

Let $a \in B \Rightarrow a = r + I$

$\Rightarrow$ there is no $p \in LT(G) suchthat\ p/r$

since r is a term, $r \in T(R)$ we have $r \in RT(I)$ such that $Q(r) = r + I = a \in B$

$\therefore$ Q is bijective.

By Lemma 5.1.6. and Proposition 5.1.7. it is straight forward to deduce the following lemma.

Lemma 5.1.8. *[1] Let $\leq$ be a term order on $T(R)$ and G a Groebner basis of I with respect to $\leq$. Then*

$$\begin{aligned} \dim(R/I) &= num\{t \in T(R) : s \text{ does not divide } t \text{ for all } s \in LT(I)\} \\ &= num\{ t \in T(R) : s \text{ does not divide } t \text{ for all } s \in LT(G)\} \end{aligned}$$

Applying the this lemma to Groebner basis with univariate head terms yield the following corollary:

Corollary 5.1.9. *[1] Let $G = \{g_1, ..., g_n\}$be a Groebner basis for the ideal $I \subset R = F[x_1, ..., x_n]$ with head terms $x_1^{d_1}, ..., x_n^{d_n}$. Then* $\dim(R/I) = d_1.....d_n$.

Proof:

by the lemma 4.1.8. we have

$$\begin{aligned} \dim(R/I) &= num\{t \in T(R) : s \text{ does not divide } t \text{ for all } s \in LT(I) \\ \} &= num\{t \in T(R) : x_i^{d_i} \text{ does not divide } t \text{ for all } s \in LT(I)\} \end{aligned}$$

let $t = \prod x_i^{c_i}$ and $x_i^{d_i}$ does not divide $t \forall\ i = 1, ..., n$

$\Rightarrow\ c_i < d_i\ \forall\ i = 1, ..., n$

$\Rightarrow$ any term in $T(R)$ with $t = \prod x_i^{c_i}$ and $c_i < d_i\ \forall\ i = 1, ..., n$ is in$\{t \in T(R) :$ $x_i^{d_i}$ *does not divide* t *for all* $s \in LT(I)$

$\Rightarrow num\{t \in T(R) : x_i^{d_i}$ *does not divide* t *for all* $s \in LT(I)\} = \prod d_i$.

This result is sufficient to give a bound on the complexity of the Groebner basis conversion using FGLM. From the above Corollary we conclude that the vector space dimension of the ideal by the Groebner basis Λ is way too big for the FGLM algorithm be useful for cryptanalytic purpose in this case

$$\dim(R/\Lambda) = 254^{200} \approx 2^{1600}.$$

5.2 Elimination of Variables

In this section we establish that the dimension of the vector space of the ideal remains invariant when eliminating certain variables. We first prove the following more general statement:

Proposition 5.2.1. *Let I' be a zero-dimensional ideal of $R' = F[x_1, ..., x_n]$, I an ideal of $R = R'[x_{n+1}]$ and $I^{'} = I \cap R^{'}$. Then* $\dim(R/I) = dim(R'/I')$ *iff there exists a polynomial $g \in R' such that x_{n+1} + g \in I$.*

Proof:

W.L.O.G. we fix a lexicographical term ordering such that x_{n+1} is the greatest variable. Let $RT(I)$ and $RT(I')$ be defined as follows:

$RT(I) = t \in T(R) : s does\ not\ divide t \forall s \in LT(I)$

$RT(I) = t \in T(R') : s does\ not\ divide t \forall s \in LT(I') \subset RT(I)$

By the above Lemma, $\dim_k(R/I) = numRT(I)$ holds. Thus it is sufficient to prove that $numRT(I = RT(I'))$.

Since x_{n+1} does not divide t for $t \in T(R')$, the equality $RT(I) = RT(I')$ holds iff $x_{n+1} \in LT(I)$, that is there exists a $g \in R'$ for which $x_{n+1} + g \in I$.

So even eliminating a significant amount of variables does not reduce the complexity of converting the Groebner basis to a term order suitable for key recovery.

Testing Keys

Groebner bases were invented to solve the ideal membership problem. So why are we not able to simply test whether a linear polynomial of the form

$$k_i + C, C \in F \tag{5.2.1}$$

with C being a key variable guess, lies in the ideal? After all, this would allow us to determine the *key* piecementally by guessing each byte.

Several problems present themselves here. First of all, the polynomial system has solutions over the closure of the ground field, which means that we have to test for a

polynomial

$$g = p.\prod(k_i + C_j)_{t_j}, t_j \in \mathbb{N}_{\not\vdash}, C_j \in F$$

instead, where the C_j denote candidate values for the key variable and p is a product of irreducible non-linear polynomials.Moreover the dimension of the ideal again plays an important role here: it is an upper bound on the number of solutions of the corresponding polynomial system in the closure of the field. Hence the degree of g is expected to be very large.

Chapter 6

Conclusion

We have demonstrated, AES in detail, change the key recovery problem of AES-128 in to a zero-dimensional Groebner ideal with out any polynomial reduction and try to implicate the finding result. During the process we explain how to get the zero-dimensional Groebner basis, just by choosing a particular variable order and a term order, degree lexicographic. But, in order the result to have a meaning full cryptanalytic sense we had try to convert the the Groebner basis to a Groebner basis with suitable term order. We also choose an algorithm for the Groebner basis conversion, FGLM, which works only for zero-dimensional ideal. And, the algorithm has its own shortage, the time complexity of the algorithm to change a Groebner basis of high degree like in our case is very big. All in all we have learned that the approach we had followed do not translate in to successful cryptanalysis. It is an open problem whether the result contained in this thesis can be leverage into an attack. And, the writer believes:

1. developing an other algorithm, used for Groebner basis conversion, which would have a less time complexity,

2. constructing a Groebner basis for the key recovery problem by a more suitable

term order and analyse the result or develop a universal Groebner basis for the key recovery problem and analyse the result.

Bibliography

[1] Thomas Becker and Volker Wiespfenning. *A computational Approach to Commutative Algebra.* Springer-Yerlag, 1991.

[2] Philip Benge, Valerie Burks, Nicholas Cobar, *Groebner Basis Conversion Using FGLM Algorithm.*

[3] Johannes Buchmann, Andrie Pyshkin, Ralph-Philipp Weinmann. *A zero Dimensional Groebner basis for AES-128.*

[4] Stephane Collart, Micheal Kalkbrene, and Daniel Mall. Converting Bases with the Groebner Walk, *Journal of Symbolic Computation*, 24(3/4):465-469, 1997.

[5] Nicolas Courtois and Josef Pieprzyk. *Cryptanalysis of Block Ciphers with Over defined Systems of Equations.* In Yuliang Zheng, editor, Advances in Cryptology ASIACRYPT 2002, volume 2501 of Lecture Notes in Computer Science, pages 267287. Springer, 2002.

[6] David A. Cox, John B. Little, and Don O'Shea. *Ideals, Varieties, and Algorithms.* Springer-Verlag, NY, 2nd edition, 1996. 536 pages.

[7] Hans Delfs, Helmut Knebl. *Introduction to Cryptography Principle and application.* Springer-Verlag Berlin Heidelberg, 2nd edition 2007.

[8] Jean-Charles Faueger. *A New Efficient Algorithm for computing Groebner Bases (F4).* Journal of Pure and Applied Algebra, 1999.

[9] William Stallings. *Cryptography and Network Security Principles and Practices.* Prentice Hall, 4th edition, 2005. 592 pages.

[10] Mark Stamp, Richard M. Low. *Applied Cryptanalysis, Breaking Ciphers in Real World.* Wiley-InterScience A John Wiley and Sons.INC Publication.

[11] National Institute of Standards and Technology. *FIP-197: Advanced Encryption Standard*, November 2001.