Rijndael Circuit Level Cryptanalysis

by

Serdar Pehlivanoglu

A Thesis
Submitted to the Faculty
of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the
Degree of Master of Science
in

Computer Science

by

April, 2005

Approved:

Dr. William J. Martin Thesis Advisor CS Department	Dr. Berk Sunar Thesis Committee ECE Department
Dr. Stanley Selkow Thesis Committee CS Department	Dr. Michael Gennert Department Head CS Department

Abstract

The Rijndael cipher was chosen as the Advanced Encryption Standard (AES) in August 1999. Its internal structure exhibits unusual properties such as a clean and simple algebraic description for the S-box. In this research, we construct a scalable family of ciphers which behave very much like the original Rijndael. This approach gives us the opportunity to use computational complexity theory. In the main result, we generate a candidate one-way function family from the scalable Rijndael family. We note that, although reduction to one-way functions is a common theme in the theory of public-key cryptography, it is rare to have such a defense of security in the private-key theatre.

In this thesis a plan of attack is introduced at the circuit level whose aim is not break the cryptosystem in any practical way, but simply to break the very bold Rijndael security claim. To achieve this goal, we are led to a formal understanding of the Rijndael security claim, juxtaposing it with rigorous security treatments. Several of the questions that arise in this regard are as follows: "Do invertible functions represented by circuits with very small numbers of gates have better than worst case implementations for their inverses?" "How many plaintext/ciphertext pairs are needed to uniquely determine the Rijndael key?"

Preface

In this thesis I describe the research I conducted in pursuit of my Master's Degree in Computer Science Department at WPI.

I would first like to thank Prof. William Martin, who has been my advisor and mentor. I'm more than grateful to him for the advice, guidance, trust and the funding he has provided with me. I feel honered by being able to work with him.

I'm very grateful to the members of my thesis committee, Prof. Berk Sunar and Prof. Stanley Selkow, for their support, advice and time since I forced them with tight schedules. I want to give special thanks to Prof. Sunar, who helped me tremendously with all my questions on cryptography.

I would like to thank Kaan Yuksel, Selcuk Baktir, Jens Peter Kaps, Gunnar Gaubatz and Erdinc Ozturk for such a great atmosphere in and surrounding the Cryptography and Information Security Lab.

Last but not least, I would like to thank my parents and my friends for being patient with me and for their support during my graduate studies.

This M.S. Thesis is based upon work supported by the National Science Foundation under Grant No. ANI-0112889.

Serdar Pehlivanoglu

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cryptography has been used almost since people had some written object to obscure. However, modern cryptography had expanded its field by covering related problems and goals. Those new goals require more than keeping privacy of communications; such as digital signatures or digital cash. Moreover, it became an important issue to authenticate parties communicating with each other. The creation of randomness is another topic related to modern cryptography. Not only these, but also modern cryptography shifted the focus from the art of the cryptography to the science of it.

This discipline mainly remained an art before Claude Shannon pointed out the science in his famous work [Sh49]: "As a first step in the mathematical analysis of cryptography, it is necessary to idealize the situation suitably, and to define in a mathematically acceptable way what we shall mean by a secrecy system."

Shannon distinguishes two types of security: (1) Theoretical and (2) Practical. Theoretical security, which is named as 'unconditional', is the one against an adversary who has unlimited time and computational resources. Practical (computational) security is resistance to an attack using a specified limited amount of time and computational resources. Although it requires a mathematical theory for cryptography to ensure provably computationally-secure systems, to the best of our knowledge there is not much progress to ensure secrecy of systems 'in a mathematically acceptable way'. To quote James L. Massey [Ma04a]: "Today almost everyone plies the art of cryptography, generating more and more schemes that nobody can prove are computationally secure."

1.1 Public Key vs. Private Key

Private key cryptography is the most commonly used method to communicate securely. In this scheme, the sender and receiver ensure secrecy by use of a private key on which they agree before the communication starts. The sender encrypts the message with this secret key and sends it via an some insecure channel which is subject to attack by an adversary. The receiver is able to decrypt the transmitted message with the same secret key. The security of the scheme is based on its resistance to an adversary who knows the cryptosystem (by Kerckhoff's principle) and who has the ability to eavesdrop on the insecure channel.

Symmetric key (private key) ciphers can be grouped broadly into block ciphers and stream ciphers. A block cipher operates on a group of bits, in contrast to a stream cipher, which encrypts one bit at a time. Stream ciphers are generally faster than block ciphers in hardware, and have less complex hardware circuits. However, block ciphers are the fundamental and the most prominent elements since they support message authentication techniques, data integrity and cryptographic hash functions.

The problem in the symmetric key setting is that the key must be exchanged in a secure way, and not by the means of normal communication. In 1976, Whitfield Diffie and Martin Hellman developed the concept of asymmetric public-key cryptography [DiHe76] to handle this problem. In this scheme, there are two keys used; public-key and private-key, with public key for encryption and private key for decryption. These keys are mathematically related to each other but it is hard to derive either key from the other. The public key can be sent out over an insecure channel, and any message directed to the owner of this public key can be encrypted by using this publicly known key. However, since it's hard to find the private key, only the owner is able to decrypt transmitted messages. In fact, the private key is kept by the owner of the corresponding public-key to decrypt messages.

In this way, one could send a secret message without ever meeting with the receiver, and only the intended receiver will understand the content of the message. Therefore, it is also easy to handle key-distribution problem, since one doesn't have

to agree on a secret key before the communication itself. All the sender does is just look up some information, which is publicly known to anybody else, and then use this information to encrypt any message.

1.2 Security Issues

The goal of a security system is that an adversary should not be able to learn anything about the plaintext if he is given a ciphertext. However, there is not a formal way to understand what that means, in other terms we do not have a 'mathematically acceptable way' to define secrecy. For example, in the symmetric key system, the length of key is n bits, then we have a probability of at least $1/2^n$ to find the correct key (just by random selection), and to get all the information on the channel unless the key is changed. This doesn't make the scheme bad, but it doesn't either show that the scheme is secure.

From Shannon's ideas to today's security concerns, it is believed that the best way to design a cipher is to construct it in such a way that breaking the cipher turns out to solve a difficult problem. This belief is also basic assumption in the provable security treatment, about which we gave introductory detail in Section 2.3.

Since it is so much important to find difficult problems, the question of hardness is another issue which cryptographers have to come with 'mathematically acceptable way'. Public-key cryptosystems are usually based on 'hard' problems on the avarage

case. Computational complexities of problem sets and complexity classes are therefore central to the discussion of public key security. However, symmetric setting, mainly block ciphers have different security concerns.

First of all, it is not natural to use computational complexity theory, since this discipline doesn't cover the instances of problem sets such as secrecy of a block cipher. On the other hand, all we have on security of block ciphers are just fixed lists of attack strategies and the claims regarding the resistance of ciphers against these strategies. So, basically we have a game of ad hoc cryptanalysis on the security of block ciphers. Chapter 2 enhances the discussion on security of block ciphers.

1.3 AES Competition

Approved in 1977, Data Encryption Standard(DES) was an early standard for private-key cryptography. On January of 1997, National Institute of Standards and Technology (NIST) called for cryptographers to propose a new standard block cipher for United States Government use in non-classified but sensitive applications. The Advanced Encryption Standard (AES) was supposed to replace Triple-DES which is a slight modification of the original DES. The main reasons for this major change were the DES itself was vulnerable to the exhaustive search and Triple-DES was relatively slow in software.

The winner of the contest, where the whole crypto community was invited to

evaluate candidates, would become the standard for NIST, ISO, IETF and IEEE. After a long process, in August 1999, five finalists were announced. The NIST's fact sheet explains why the Rijndael Cipher of Vincent Rijmen and Joan Daemen was chosen out of those finalists: "When considered together, Rijndael's combination of security, performance, efficiency, ease of implementation, and flexibility makes it an appropriate selection for the AES. Specifically, Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes."

1.4 Contribution of the Thesis

In this thesis, a plan of attack on the Rijndael circuit is introduced. To make progress on this plan, a broad range of discussions are made about the gate complexity theory and the block cipher security. All of the work explained in this thesis is motivated by the belief that a mathematical theory of cryptography should be possible.

Since the aim of this attack plan is to break the Rijndael security claim (RSC), a formal understanding of the RSC is sought. We indicated that it is reasonable to relate the RSC to being finite pseudorandom permutation family (FPRP) which was introduced by Mihir Bellare and Phillip Rogaway.

We have claimed that the proposal of Rijndael allows a well-defined block cipher family which behaves like the original Rijndael. Hence, a scalable Rijndael family is

specified to provide a circuit level cryptanalysis by use of computational complexity theory.

One of the most significant contributions of this thesis is generating a one-way function family from the scalable Rijndael family. This conclusion holds true if the RSC is proven for block ciphers in the family, whose parameters are sufficiently large. Remark that, although reduction to one-way functions is a common theme in the theory of public-key cryptography, it is rare to have such a defense of security in the private-key theatre. This is mainly the consequence of the Rijndael proposal which supports block cipher family.

The above discussion deals with topics related to the attack plan. The main step of the attack plan, introduced in the thesis, is seeking answers for the questions arised in this regard: "Do invertible functions represented by circuits with very small numbers of gates have better than worst case implementations for their inverses?" "How many plaintext/ciphertext pairs are needed to uniquely determine the Rijndael key?" In this thesis, a methodology is given to elaborate on the gate complexity theory, about which James L. Massey believes that it helps to determine the difficulty of a problem as part of a mathematical theory of cryptography for computational security.

1.5 Outline of the Thesis

Chapter 2 discusses the security concerns of block ciphers in great detail. Furthermore, it explains the Rijndael Security Claim and the conditions under which the cipher fails to satisfy this claim. According to our mathematical analysis, we need a formal notion what is meant by RSC, so in this chapter we also describe the provable security treatment and relate its assumptions to the RSC.

Chapter 3 starts with introducing our motivations on defining scalable Rijndael family and motivations on circuit level cryptanalysis. It continues with a possible attack on the RSC. In this chapter, we will clearly introduce the main steps of such an attack and the implications of success.

The following four chapters are reserved for the main steps of our attack plan. Chapter 4 introduces a scalable Rijndael family along with its gate complexity. Chapter 5 questions the existence of poly-sized plaintext-ciphertext pairs to determine the key. Also, we will generate one-way function family which is directly dependent on the proof of RSC. Chapter 6 is about inverting circuits and the question whether small circuits have relatively small inverse circuits.

Finally, in Chapter 7 a summary of work that has been done is given with possible future research on this topic.

Chapter 2

Rijndael Security Claim

Symmetric encryption allows two parties to communicate securely by use of a private key. The bits transmitted between parties are produced by processing data as a function of this shared key. The encryption scheme also specifies a decryption algorithm to retrieve the original data from the transmitted one.

Block ciphers are the most popular and the easiest way of achieving symmetric encryption. As an example, we could name the Data Encryption Standard (DES) which was used from 1974 until recently. It is being replaced by the Advanced Encryption Standard, namely Rijndael, which is already in widespread use.

A block cipher is an encryption function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ that takes two inputs, a k-bit key K and an n-bit plaintext M and returns an n-bit ciphertext $C = E(K, M)$. For each key $K \in \{0,1\}^k$ we let $E_K : \{0,1\}^n \to \{0,1\}^n$

be the function defined by $E_K(M) = E(K, M)$. For any block cipher and any key K, the function E_K is a permutation on $\{0,1\}^n$. Since it is a permutation, we let $E^{-1} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ be defined by $E^{-1}(K, C) = E_K^{-1}(C)$, which is the decryption function.

Demonstrating existence of a key recovery attack, with all possible modes of access for the adversary (known/chosen/adaptively chosen plaintext/ciphertext, known/chosen/ adaptively chosen key relations) is a way to show that a given block cipher fails to be secure. Any secure block cipher must be strong enough such that it shouldn't be possible to find a key-recovery attack. But this is not all that we seek from a secure encryption scheme.

Consider an encryption scheme E that uses AES as an internal algorithm: $E : \{0,1\}^{128} \times \{0,1\}^{256} \to \{0,1\}^{256}$ defined by $E_K(M_1, M_2) = (M_1 \oplus M_2, AES_K(M_1 \oplus M_2))$. Key recovery is as hard as it is for AES, but the scheme leaks information when an adversary is given the ciphertext. This encryption scheme is not acceptable, since $M_1 \oplus M_2$ is leaked.

This example shows the insufficiency of defining security in terms of key-recovery alone. So what makes a symmetric encryption system secure? This question has on occasion led to the proposal of huge lists of security conditions to be satisfied by a scheme, many of which are practically impossible to check.

Historically, there seem to be few attempts to give a methodology for checking

security. There are some common attack strategies on block ciphers, and the designers of a cipher usually come up with conjectures on how their encryption scheme is secure against well-known attack strategies. We see that Rijndael's authors wanted to fill that gap practically in their AES proposal in 1999. Their security treatment, which will be explored later, is a clever and easy way of claiming security. However, they did not provide rigorous statements to specify what they mean.

Since 1994, the work of Phillip Rogaway and Mihir Bellare on provable security has answered the theoretical issues in security and has specified almost theoretically complete security treatment for symmetric encryption schemes. Their 2003 RSA Conference Award in mathematics for their work on practice-oriented provable security suggests that their security treatment is accepted by the community and promises extended use for the future.

In this chapter, we want to give an outline of the Rijndael encryption algorithm, referring the reader to AES Proposal ([DR99]), the book "The Design of Rijndael" ([DR01]), official publication standard from NIST ([NIST01]) and a paper ([Gl02]) of Brian Gladman written to revise AES proposal for more specification of Rijndael. Then we will briefly discuss the Rijndael security claim (RSC). After introducing RSC, we want to juxtapose it with rigorous concepts of security.

2.1 Rijndael Specification

Rijndael operates on a 4×4 array of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state). For encryption, each round of Rijndael (except the last round) consists of four stages.

Following Figure 2.1 is the schematic representation of Rijndael:

2.1.1 The SubBytes Transformation

We took the description of SubBytes transformation from [NIST01]:

> The *SubBytes()* transformation is a non-linear byte substitution that operates independently on each byte of the state using a substitution table (S-box). This S-box, which is invertible, is constructed by composing two transformations:
>
> 1. Take the multiplicative inverse in the finite field $GF(2^8)$ modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$; the element 00 is mapped to itself.
>
> 2. Apply the following affine transformation (over $GF(2)$):
>
> $$b'_i = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i$$
>
> for $0 \leq i < 8$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value {63} or {01100011}. Here and elsewhere,

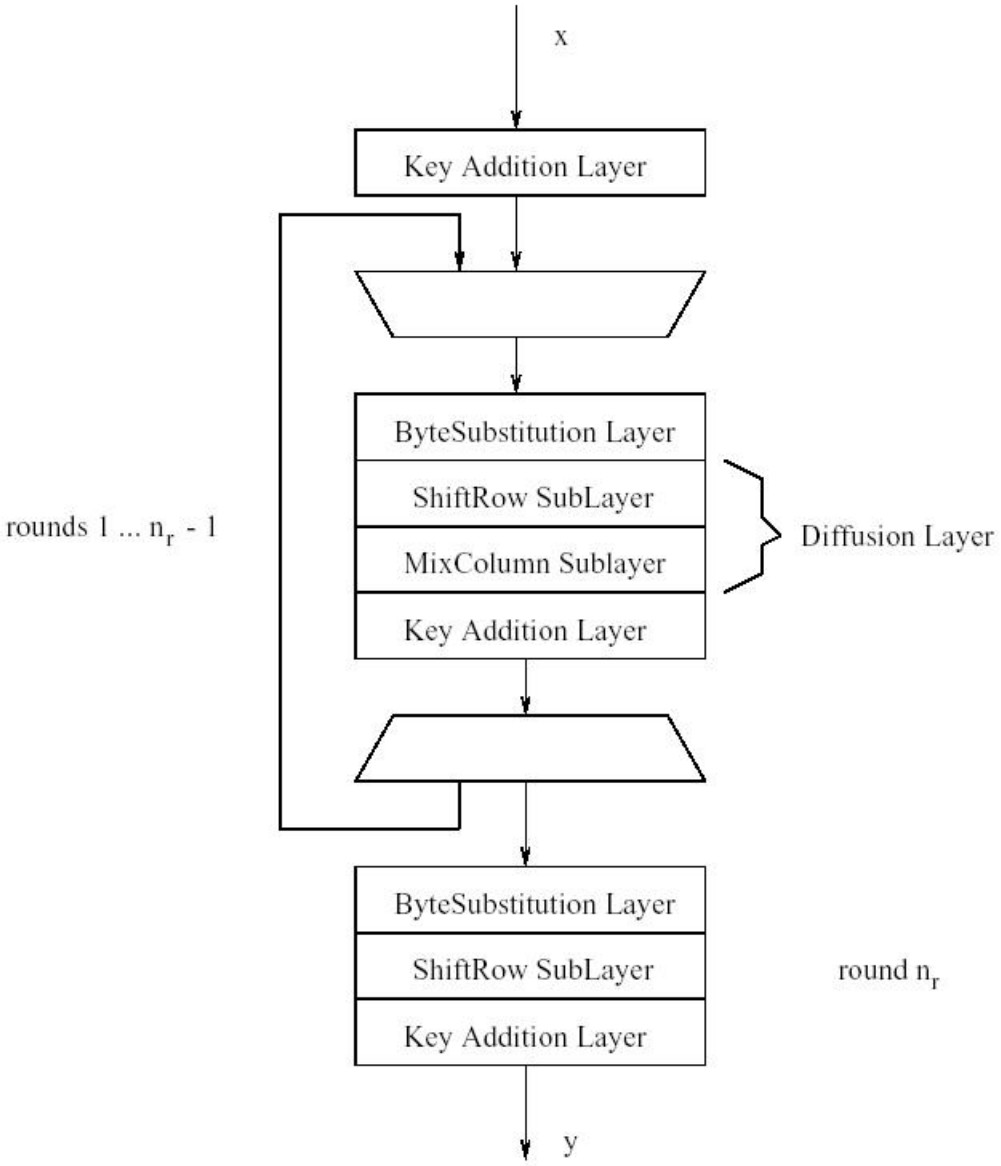

Figure 2.1: [LNCP]Rijndael Cipher

a prime on a variable (e.g., b') indicates that the variable is to be updated with the value on the right.

Figure 2.2 illustrates the effect of the SubBytes() transformation on the state.

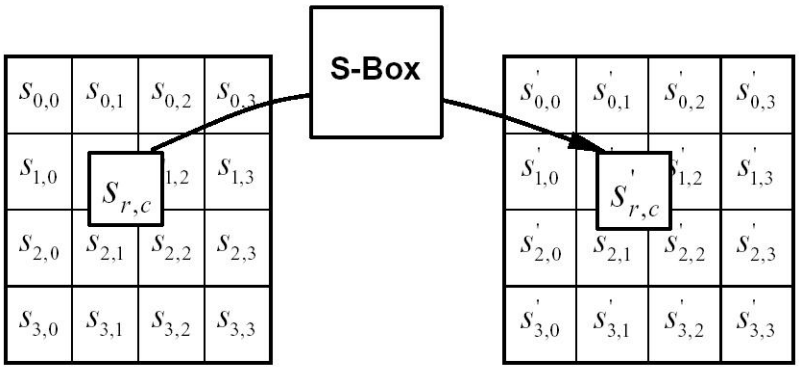

Figure 2.2: [NIST01]*SubBytes()* applies the S-box to each byte of the state

We describe inverse operation as [DR01] explains:

Inverse Operation: The inverse operation of *SubBytes* is called *InvSubBytes*. It is a bricklayer permutation consisting of the inverse S-box S_{RD}^{-1} applied to the bytes of the state. The inverse S-box S_{RD}^{-1} is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in $GF(2^8)$.

2.1.2 The ShiftRows Transformation

The definition of Shiftrows transformation and its inverse are directly taken from [DR01]:

> The *ShiftRows* step is a byte transposition that cyclically shifts the rows of the state over different offsets. Row 0 is shifted over C_0 bytes, row 1 over C_1 bytes, row 2 over C_2 bytes and row 3 over C_3 bytes, so that the byte at position j in row i moves to position $(j - C_i) \pmod{N_b}$. The shift offsets C_0, C_1, C_2 and C_3 depend on the value of N_b.

Figure 2.3 illustrates the effect of the ShiftRows step on the state.

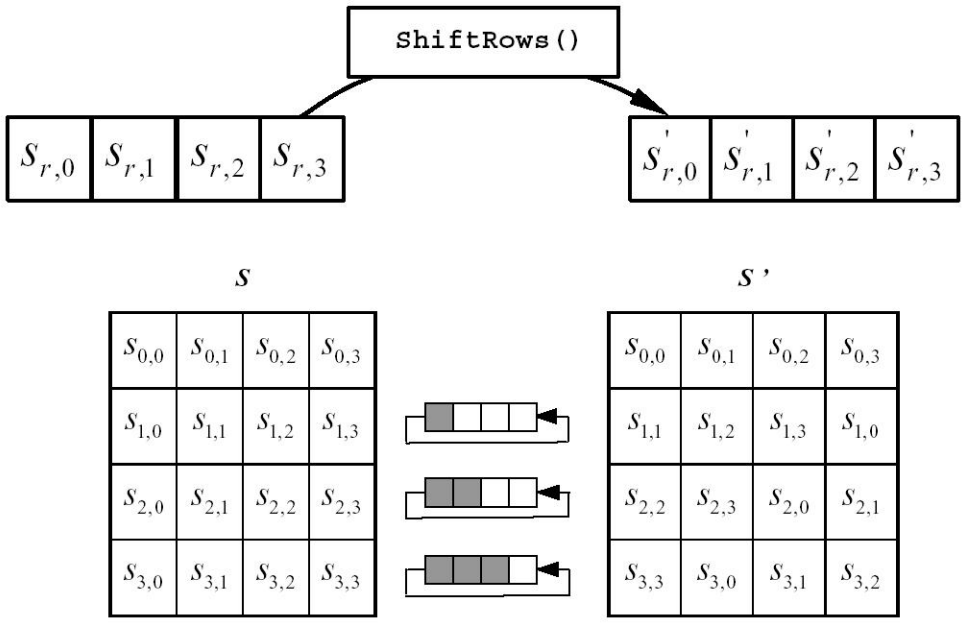

Figure 2.3: [NIST01]ShiftRows cyclically shifts the last three rows in the state.

Inverse Operation: The inverse operation of *ShiftRows* is called *InvShiftRows*. It is a cyclic shift of the 3 bottom rows over $N_b - C - 1$, $N_b - C_2$ and $N_b - C_3$ bytes respectively so that the byte at position j in row i moves to position $(j + C_i) \pmod{N_b}$.

2.1.3 The MixColumns Transformation

The definition of MixColumns transformation and its inverse are directly taken from [DR01]:

The *MixColumns* step is a bricklayer permutation operating on the state column by column. The columns of the state are considered as polynomials over $GF(2^8)$ and multiplied modulo x^4+1 with a fixed polynomial $c(x)$. The polynomial $c(x)$ is given by (+ and $\cdot$ are arithmetic operations in GF):

$$c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$$

This polynomial is coprime to $x^4 + 1$ and therefore invertible. The modular multiplication with a fixed polynomial can be written as a matrix multiplication. Let $b(x) = c(x) \cdot a(x) \pmod{x^4 + 1}$. Then

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figure 2.4 illustrates the effect of the MixColumns step on the state.

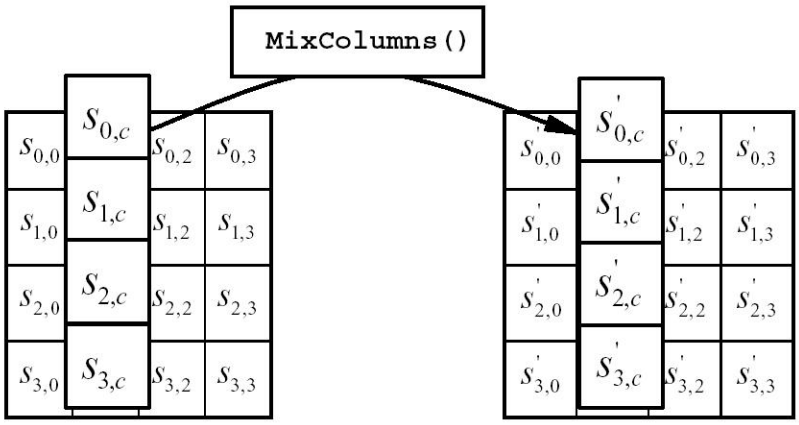

Figure 2.4: [NIST01]MixColumns operates on the state column-by-column.

Inverse Operation: The inverse operation of *MixColumns* is called *InvMixColumns*. It is similar to MixColumns. Every column is transformed by multiplying it with a fixed multiplication polynomial $d(x)$, defined by

$$(03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02).d(x) \equiv 01 \pmod{x^4 + 1}$$

it is given by

$$d(x) = 0B \cdot x^3 + 0D \cdot x^2 + 09 \cdot x + 0E.$$

Written as a matrix multiplication, InvMixColumns transforms the columns in the following way:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \times \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

2.1.4 XorRoundKey Transformation

The definition of XorRoundKey transformation is directly taken from [DR01]:

> The key addition is denoted *XorRoundKey*. In this transformation, the state is modified by combining it with a round key with the bitwise XOR operation. A round key is denoted by *ExpandedKey[i]*, $0 \leq i \leq N_r$. The array of round keys *ExpandedKey* is derived from the cipher key by means of the key schedule. the round key length is equal to the block length. The *XorRoundKey* transformation is illustrated in Figure 2.5. XorRoudKey is its own inverse.

2.1.5 The Key Schedule

The following description of *Key Schedule* is directly taken from [Gl02]:

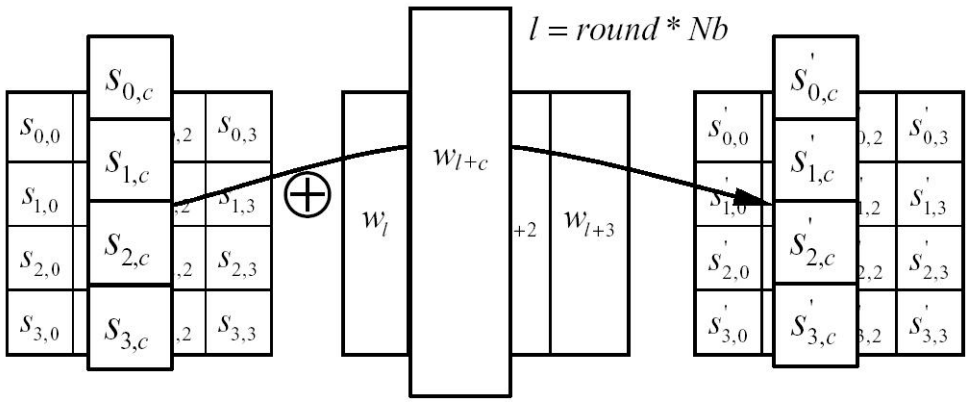

Figure 2.5: [NIST01]XorRoundKey XORs each column of the state with a word from the key schedule.

The round keys are derived from the cipher key by means of a key schedule with each round requiring N_c words of key data which, with an additional set, makes a total of $N_c(N_r + 1)$ or as a two dimensional array k[n,c] of $N_r + 1$ round keys, each or which individually consists of a sub-array of N_c words.

The expansion of the input key into the key schedule proceeds according to the following pseudo code. The function *SubWord(x)* gives an output word for which the S-box substitution has been individually applied to each of the four bytes of its input x. The function *RotWord(x)* converts an input word $[b_3, b_2, b_1, b_0]$ to an output $[b_0, b_3, b_2, b_1]$. The word array *Rcon[i]* contains the values $[0, 0, 0, x^{i-1}]$ with x^{i-1} being the powers

of x in the field $GF(256)$.

```
KeyExpansion(byte key[4*Nk], word k[Nr+1, Nc], Nc, Nk, Nr)

 begin
     i=0
     while (i < Nk)
         k[i] = word [key[4*i+3], key[4*i+2],key[4*i+1],key[4*i] ]
         i = i+1
     end while


     i = Nk
     while (i < Nc * (Nr+1))
         word temp = k[i-1]


         if (i mod Nk = 0)
             temp = SubWord(RotWord(temp)) xor Rcon[i / Nk]
         else if ((Nk > 6) and (i mod Nk=4))
             temp = SubWord(temp)
         end if


         k[i] = k[i - Nk] xor temp
         i = i+1
     end while
 end
```

2.1.6 The Decryption Function

In description of each transformation, we already gave how inverse operation works. In this section we provide following pseudo code for the decryption function from [Gl02]:

```
InvCipher(byte in[4*Nc], byte out[4*Nc], word k[Nr+1, Nc], Nc, Nr)

Begin
    byte state[4, Nc]
    state = in
    XorRoundKey(state, k[Nr*Nc..(Nr+1)*Nc-1], Nc)
    for round =Nr-1 step -1 to 1
        InvShiftRows(state, Nc)
        InvSubBytes(state, Nc)
        XorRoundKey(state, k[round*Nc..(round+1)*Nc-1])
        InvMixColumns(state, Nc)
    end for
    InvShiftRows(state, Nc)
    InvSubBytes(state, Nc)
    XorRoundKey(state, k[0..Nc-1])
    out = state
end
```

2.2 Rijndael Security Claim

To understand the strength of a block cipher, comparing it with majority of ciphers is a nice idea. This comparison is the basis of the security criteria introduced by Joan Daemen and Vincent Rijmen [DR01].

To understand this criterion let's determine the number of all possible block ciphers of dimensions n_b (block length) and n_k (key length). If the key is fixed then the number of inputs is 2^{n_b}, so the cipher behaves as one of the $2^{n_b}!$ permutations. If this is the case for each key, then the number of all possible block ciphers is $2^{n_b}!^{2^{n_k}}$.

This notion of security seems based om the assumption that the majority of ciphers are not vulnerable to an exploitable weaknesses since the possible block ciphers in the same dimension cover all permutation families. Otherwise it would be impossible to talk about secure encryption schemes. Indeed, this must be what motivates definitions of *K-secure* and *hermetic* as security criteria in [DR01].

Definition 2.2.1 (DR01) *A block cipher is* **K-secure** *if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible block ciphers with the same dimensions. This must be the case for all possible modes of access for the adversary (known/chosen/adaptively chosen plaintext/ciphertext, known/chosen/adaptively chosen key relations . . .) and for any a priori key distribution.*

This definition is very strong in terms of security. So according to [DR01], if one finds any key-recovering attack faster than the exhaustive search, symmetry properties, non-negligible class of weak keys or related-key attacks, then the cipher fails to satisfy this criteria. Here are some brief explanations about these kind of attacks:

1. **Existence of a key-recovering attack faster than exhaustive search:** There is a finite set of possible keys in a block cipher system. One can go through all these possible keys which is called as exhaustive search. Because of the exponential complexity of block cipher, this kind of brute force attack is impractical. So an opponent seeks out better key-recovery attacks. According to *Kerckhoff*'s principle, we assume that the opponent knows the cryptosystem being used. Following are the most common types of attack schemas, for classifying attacks based on resources available to the attacker, discussed in [Sti95]:

    - Ciphertext-only: The opponent possesses a string of ciphertext, y.
    - Known plaintext: The opponent possesses a string of plaintext, x, and the corresponding ciphertext y.
    - Chosen plaintext: The opponent has obtained temporary access to the encryption machinery. Hence he can choose a plaintext string, x, and construct the corresponding ciphertext string, y.
    - Chosen ciphertext: The opponent has obtained temporary access to the

decryption machinery. Hence he can choose a ciphertext string, y, and construct the corresponding plaintext string, x.

2. **Finding symmetry properties:** The behavior of a block cipher may exhibit some symmetries. Those kind of symmetries give some strength to opponent. Consider such an extreme property for virtually all plaintexts: if the corresponding ciphertext of any plaintext x is the binary complement of the ciphertext corresponding to binary complement of x, then at least, one can define an attack with at most half the complexity of exhaustive search.

3. **Finding non-negligible class of weak keys:** Weak keys are secret keys for which the block cipher exhibits certain regularities. For example in DES there are four keys for which encryption is exactly the same as decryption. This means that if one were to encrypt twice with one of these weak keys, then the original plaintext would be recovered. For IDEA, there is a class of keys for which cryptanalysis is easy to recover key. For any block cipher, there might as well be a large set of weak keys (perhaps even with the weakness exhibiting itself in a different way) for which the chance of picking a weak key is too large for comfort. In such a case, the presence of weak keys would have an obvious impact on the security of the block cipher.

4. **Related-key attacks:** In this kind of attack scenario, the attacker knows (or

chooses) a relation between several keys and he can access to the encryption function using such related keys. The goal of the attacker is to find the keys themselves. The scenario of the attack is very powerful in terms of the attacker's capabilities and thus quite unrealistic in practice.

Sometimes while a given cipher is K-secure, if you use it as a component in a larger scheme, any weakness in the cipher may cause troubles to the larger scheme. This motivates the following definition:

Definition 2.2.2 (DR01) *A block cipher is* **hermetic** *if it does not have weaknesses that are not present for the majority of block ciphers with the same block and key length.*

The security claims of Rijndael are that the Rijndael cipher is K-secure and hermetic.

2.3 Juxtaposition Of Claim

In the previous section, we explained the Rijndael Security claim and elaborated on the authors' understanding of security with their definitions. Most of the time, it becomes an easy question to answer what makes a block cipher fail to be secure instead of showing how secure it is. This is the reason why the authors of the Rijndael focused on "security failure". In this section we want to briefly introduce provable

security, then juxtapose the Rijndael's security claim with the basic concepts of provable security.

The idea of provable security was introduced in the pioneering work of Goldwasser and Micali [GoMi84]. They developed it in the particular context of asymmetric encryption, but it affected the understanding of security, as well as, security failure and was soon applied to other tasks. However, with appropriate paradigm shifts, the idea of provable security is made useful for some other applications, such as block cipher security.

I refer the reader to [Be98] for an outline of provable security. But simply put, the goal of provable security is proving that the only way to defeat a cryptographic protocol is to break its underlying atomic primitive. Some examples of atomic primitives are those belonging to block ciphers or the RSA problem or the NTRU problem. The distinction between protocols and atomic primitives is that in their purest and rawest state, atomic primitives don't solve any cryptographic problem we actually care about. The goal of provable security is achieved by a proper reduction from the security of an atomic primitive to the security of a protocol.

The idea of reduction works similar to the one we know from the theory of NP-completeness. For example, there may be a reduction from one-wayness of the RSA encryption function to the security of a protocol that uses this RSA function as a primitive. If there is a program P that breaks the protocol, then existence of reduc-

tion allows us to construct a program P' that provably breaks the RSA encryption function.

With Bellare's sentences, an important part of this framework is that in order to enable a reduction, one must also have a formal notion of what is meant by the security of the underlying atomic primitive? This is the part where work on provable security contributes to the formalization of security treatments, especially the formalization of symmetric encryption security.

There is enough information on the introduction of notions, or definitions that enable us to think about atomic primitives in a systematic way, in a set of papers authored by Bellare and Rogaway [BeRo93, 94, 97]. Lack of a rigorous framework can seem to be a big obstacle in understanding security; this became apparent to the author as he tried to work on the Rijndael security claim. Absence of a formal notion in the Rijndael security claim makes it difficult to work systematically on it. It is not an easy task to prove or disprove the reliability of any cryptosystem or to come up with strong implementable design criteria without formal notions of security.

However, this wasn't a big problem with cryptosystems directly relying on the strength of candidate one-way functions in which complexity-theoretic approaches could be applied. But in practice, block ciphers, the most popular atomic primitives especially for private key cryptography, were lack of any security treatment. Then one may agree that the best thing to say about security (without any formalization)

would be the one Rijndael authors already said on being K-secure and hermetic. Now I will discuss Bellare and Rogaway's formalization for block ciphers and show the relationship between their notions and the Rijndael Security Claim.

According to Bellare and Rogaway an "ideal" encryption, which is secure in the strongest possible natural sense, would be as: "An angel took the message M from the sender and delivered it to the receiver in some magical way". The adversary would see nothing at all, and in particular, no partial information would leak. By contrast, if we appey to Kerchoff's principle, in any cryptosystem, the block length is known to the adversary a priori.

Formally modeling block ciphers to achieve ideal encryption leads Bellare and Rogaway to model a block cipher as a finite pseudorandom permutation (FPRP) family [BeRo94]. Note that the fundamental notion of a pseudorandom function family is due to Goldreich, Goldwasser and Micali [GoMiGo86]. Loosely speaking the model requires that as long as you don't know the underlying key, the input-output behavior of a block cipher closely resembles that of a random permutation.

To check the randomness of a given block cipher, these authors came up with the idea of indistinguishability and different types of security notions such as left-or-right, real-or-random indistinguishability. They introduced a totally impractical but theoretically useful random oracle model. Leaving the details in [BeRo93, 94, 97], the basic idea behind indistinguishability (in particular real-or-random indistin-

guishability) is to consider an adversary who chooses a message to be processed by a random oracle. The oracle either responds with an encryption of this message under a randomly chosen key or responds with an output of a random function. The scheme is considered secure if the adversary has a hard time telling which alternative the random oracle responds with.

This basic idea can be extended by allowing more queries to be sent to the oracle or by changing the game for different kind of oracles (such as left-or-right oracle). In any of the notions defined by Bellare and Rogaway, there are two worlds where the adversary doesn't know in which world it is playing. The goal of the adversary is to guess in which world it is playing. The advantage of adversary is defined as the measurement of how much better than 1/2 it does at guessing which world it is in, namely the excess over 1/2 of the adversary's probability of guessing correctly.

It is better now to touch on concrete treatment of security captured by the above formalization. Rather than proving asymptotic results about the infeasibility of distinguishing the block cipher from a FPRP family, one can easily define the advantage as a function of concrete and exact parameters of the game. For example, it is possible to say "Let A be an adversary that runs in time at most t and asks at most q queries, these totaling at most σ 128-bit blocks, then A has an advantage of smaller than $c_1 \cdot \frac{t}{2^{128}} + c_2 \cdot \frac{q}{2^{128}}$."

The idea of indistinguishability is very strong in terms of covering security ac-

cording to the proposers of this idea. To quote from [LeComp]: "Our thesis is that we should consider an encryption scheme to be "secure" if and only if it is IND-CPA (indistinguishable under chosen plaintext) secure (i.e. the advantage is negligible), meaning that the above formalization captures our intuitive sense of privacy, and the security requirements that one might put on an encryption scheme can be boiled down to this one."

Remember that before the proposal of Rijndael this theory was well-designed and it seems that Rijndael security claims were inspired by this treatment and they support the design criterion of being FPRP to be a secure block cipher. Recall the definition of K-security: "A block cipher is **K-secure** if all possible attack strategies for it have the same expected work factor and storage requirements as for the majority of possible block ciphers with the same dimensions. This must be the case for all possible modes of access for the adversary (known/chosen/adaptively chosen plaintext/ciphertext, known/chosen/adaptively chosen key relations . . .) and for any a priori key distribution." Not only this, but RSC includes that Rijndael is **hermetic** which means that it doesn't have weaknesses that are not present for the majority of block ciphers with the same block and key length (Definition 2.2.2).

Regarding the above definitions, we would say that if a block cipher is K-secure and hermetic, then it is quite reasonable to believe that this block cipher behaves as a randomly chosen block cipher or, as a FPRP family. To justify this belief, let's

assume that there is an attack strategy of any kind requires not same expected work factor and storage requirements as for the majority of block ciphers (contrary to being K-secure), then this attack strategy definitely distinguishes this block cipher from a FPRP family by a game of indistinguishability conceptualized by Bellare and Rogaway. However, not always such a game distinguishes a block cipher from a FPRP family, although the block cipher may not look like random.

As an example, consider a block cipher that has pairs of keys for which the encryption is the same. Any kind of Indistinguishability Game defined and formalized under the theory of provable security may not be able to distinguish this block cipher from a FPRP family. If one can not specify an Indistinguishability game, then it is not possible to come up with a fast attack strategy compared to the majority of block ciphers. However, this block cipher still has a weakness (pairs of keys giving same encryption) that is not present for the majority of block ciphers which contradicts to be a FPRP family. Thus, if the block cipher is not hermetic although it is K-secure, then it is still not a FPRP family.

So basically, if the advantage of an adversary attacking a block cipher in any kind of Indistinguishability Game defined and formalized under the theory of provable security is not negligible, then this block cipher is not K-secure, and vice versa. If there is no such an attack strategy, but it exhibits such property that fails to satisfy being hermetic, then this block cipher is not a FPRP family. Furthermore, if it is

hermetic and K-secure then it behaves as a randomly chosen block cipher, or simply this block cipher is FPRP family.

After now we prefer to use formalization of provable security when we need a rigorous understanding of K-security and hermetic as they pertain to the RSC. However as I will elaborate our plan of attack in Chapter 3, we are not restricting ourselves to concrete security treatment and prefer to define a Scalable Rijndael Function Family.

Chapter 3

A Possible Attack Strategy

In this section, we want to give a complete description of a strategy to attack the Rijndael Security Claim (RSC). Each step we will identify contains concrete, well-defined questions. We want to investigate topics around those questions to get a better understanding of computational complexity as it relates to block cipher security, and to Rijndael in particular. We also believe that most of these questions can be attacked without need to analyze Rijndael's detailed structure. This plan of attack gives us at least basic knowledge of Rijndael circuit level analysis. On the other hand any small progress on one of the open steps would be an important result in either computational complexity or in block cipher security.

Russell Impagliazzo and Michael Luby stressed that one-way functions are essential for complexity-based cryptography [ImLu89]. In our work related to our plan of

attack, we will arrive at these same ideas, but in a very concrete way. Informally, constructing a secure block cipher, according to Rijndael Security criteria, implies the existence of one-way functions. One should beware that this plan of attack is strictly specified for Rijndael, but also can be generalized by appropriate changes if the definition of addressed block cipher allows.

3.1 Motivation

The RSC essentially says that the Rijndael block cipher behaves like a finite pseudo-random permutation family (FPRP). Existence of a meaningful attack to distinguish the block cipher from a FPRP which is better than exhaustive search is good enough to break this claim. So we are looking for exponential improvement although it may be negligible to practitioners. If the block length is n, an attack with $O(2^{n(1-c)})$ complexity where c is a constant (such as $c = 0.001$) is an exponential improvement. Its practical strength depends on constant c, but is not immediate concern to us.

Recall that the RSC compares Rijndael with all possible block ciphers in the same dimension. The idea of concrete security also suggests this kind of treatment. However, we want to interpret the Rijndael security claim via a complexity-theoretic approach. The main reason is that we are not practitioners who need numbers: "How many cycles of adversary computation can the scheme withstand?" Also Rijndael is not as other block ciphers who do not allow such a complexity theoretic approach.

Rijndael's structure allows one to generate a scalable cipher family which behaves like the original block cipher.

So basically, here are our primary reasons on the ideas presented in this work:

(i) Our goal is not cryptanalysis per se, but a better understanding of block ciphers in general. We are skeptical of RSC at the theoretical level.

(ii) We seek a fresh approach with potential to uncover weaknesses not addressed by standard cryptanalysis.

(iii) We find Rijndael appealing in its easy scalability which is a rare opportunity to consider asymptotics as key length goes to infinity.

To interpret the Rijndael security claim in complexity-theoretic language we need a scalable cipher family. We have two more reasons to construct a scalable family. (1) Security claims often include non-existence of inverting algorithms "Significantly faster than exhaustive search". For any fixed length the speed-up is constant. So we can only make this phrase precise by looking at key lengths going to infinity. We are looking for exponential improvement which is also made precise by having a scalable cipher family. (2) It seems a natural requirement for block cipher proposals with such bold security claims to include scalability of the proposal.

Assume that we have a scalable cipher family F parameterized by n, and for sufficiently large n's, the ciphers of the family F behaves like a FPRP family so that they are not distinguishable from any other block cipher in the same dimension. Then

it is possible to say that "For sufficiently large n's; let A be an adversary for F that runs in time at most $t(n)$ and asks at most $q(n)$ queries, these totaling at most $\sigma(n)$ n-bit blocks, then A has an advantage smaller than $f(n)$ where $f(n)$ is a negligible function."

We will give the technical details about our plan of attack after repeating our understanding of security one more time. As a formal notion we are saying that a block cipher should behave as a finite pseudorandom permutation family; which is also the underlying meaning of being K-secure and hermetic. So after now, when we need to question the security of any block cipher family, we will try to distinguish family members' behavior from a random permutation with same dimensions by playing the games conceptualized under provable security. We will also seek out special structure that is not present in a "majority of block ciphers".

Our attack plan is based on circuit level analysis. Cryptographers beware that each encryption algorithm has a boolean circuit implementing this algorithm, so there should be an inverse circuit somehow related to key-recovery. This is a true fact but doesn't appeal theoreticians to search for an inverse circuit. In most of the cases the problem is believed to be turned into the SAT problem which is a blind alley for theoretical computer analysis. The reason for no interest can be understood when we examine the attacks at the circuit level.

The first way of attacking is writing block cipher as a boolean circuit, and then

translating the circuit to a set of equations with one equation per boolean gate. This gives us a system of equations but not a closed algebraic formula. This would be like an instance of SAT, for which no efficient algorithm is known.

Another idea is to rewrite the formula in disjunctive or conjunctive normal forms. If one can get this simple form, the structure of the circuit would be easier to analyze. But the problem with this method is that in order to get the direct evaluation, you have to know entire plaintext/ciphertext mapping which is impossible.

Consequently, there is a series lack of literature investigating the problem from this angle. In fact, for us, this lack of circuit-level analysis in the literature is somewhat encouraging. Although we think that it is unlikely, our investigation may lead us to the SAT problem as well. On the other hand, the Rijndael circuit is very special and very fast, so it looks nothing like the general case. Moreover, it seems that we will derive from the main problem a number of interesting combinatorial challenges. Even if we are not successful in solving the challenges, it is likely that we will introduce new open problems and we will have the opportunity to address their hardness. In this way we hope to introduce new relationships between any block cipher (because our approach can be applicable to any block cipher) and such combinatorial challenges.

There is one more reason to focus on circuit level analysis. We are mostly interested with gate complexity. A gate is a boolean function of two variables. The gate complexity of a boolean function $f(x_1, x_2, ..., x_n)$ is the smallest number of gates

in an acyclic gate network that computes this function. To have a rigorous provable security treatment we need to use a complexity measure that captures the complexity of individual instances of a function. According to Massey [Ma04a], gate complexity seems to be the most useful such measure at this time. Not only by Massey, but recently gate complexity is seen to be an alternative way of quantifying security by information theoreticians. Massey argues that a mathematical theory of cryptography for computational security is not possible with the methods of number theory or theoretical computer science, but there is a chance with the methods of gate complexity (See also [Ma04b]).

We now present another argument justifying our approach using Boolean circuits. While Turing machines have by far received more attention in the computer science literature, we feel that boolean circuits are a more appropriate choice for cryptanalytic considerations. Recall that, in the non-uniform model of boolean circuit complexity, our adversary is permitted any exponential amount of precomputation so long as the results of this computation can be encoded in polynomial-sized circuits.

Let's elaborate on the above explanation. In the Turing machine model, there is a constant-sized machine processing on all input lengths n. The time complexity of the corresponding decision problem is defined as the computation time of this Turing machine on given input string. We say a Turing machine is poly-time if the computation time is bounded by some fixed polynomial of the input length. There

are some cases even though Turing machine is not poly-time but the machine could decide in polynomial time if the input length is given as an auxiliary input. That means there exists poly-time Turing machine family described independently for each input length. This is the non-uniform model of the Turing machine and each machine can be represented by poly-sized boolean circuit family.

In the non-uniform model of boolean circuit complexity it is possible that it takes a large amount of computation to find the hint (auxiliary), but once the hint is given it is possible to decide in polynomial time, since the circuit has polynomial gate complexity. Indeed, in a real-world situation involving a block cipher, a fixed ciphertext length is often known to the adversary, who can then perform a large amount of computation by knowing only the system and this block length, before attacking any specific uses of the cipher.

3.2 Attack Plan

It is now time to go further and specifically describe our plan of attack. Here are some definitions and a lemma we will use later:

Definition 3.2.1 *Let n be the block length of the Rijndael block cipher, $\mathcal{P} \subseteq F_2^n$ be the set of all possible plaintexts, $\mathcal{C} \subseteq F_2^n$ the set of all possible ciphertexts and $\mathcal{K} \subseteq F_2^n$ the set of all possible keys.*

Definition 3.2.2 *Let $\mathcal{F}$ denote the Rijndael encryption function. If $p \in \mathcal{P}$ is a plaintext and the ciphertext under the encryption of key k is c where $k \in \mathcal{K}$ and $c \in \mathcal{C}$, then we will write $\mathcal{F}_p(k) = c$.*

Lemma 3.2.1 *For any specified key value k and two different plaintexts p_1 and p_2; $\mathcal{F}_{p_1}(k) \neq \mathcal{F}_{p_2}(k)$*

Proof: Assume the conclusion is false. Then there is at least one ciphertext c, such that there is no plaintext which gives us this ciphertext. But if we decrypt this ciphertext under the decryption key which corresponds to the encryption key k, then we will have a plaintext p, which $\mathcal{F}_p(k) = c$ holds. So we can say that $\mathcal{F}$ is invertible.

♣

Definition 3.2.3 *If $P \subseteq \mathcal{P}$ is a collection of plaintexts then let's define $\mathcal{G}_P : \mathcal{K} \longrightarrow \mathcal{C}^{|P|}$. So if $k \in \mathcal{K}$ then $\mathcal{G}_P(k) = \{(p, \mathcal{F}_p(k)) \mid p \in P\}$.*

Consider specified collection of plaintexts P, and the function $\mathcal{G}_P$ over this set. If for all k_1 and k_2, there is at least one $p \in P$ such that $\mathcal{F}_p(k_1) \neq \mathcal{F}_p(k_2)$, then function $\mathcal{G}$ over the set P is invertible. That means, if one gives ciphertexts corresponding to all plaintexts in P, there is exactly one key value which generates all these plaintext-ciphertext pairs. However, we are not aware of any plaintext set so that $\mathcal{G}$ is invertible over this set. Formally we can define invertibility as follows:

Definition 3.2.4 *For a given plaintext collection P, if* $\mathcal{G}_P : \mathcal{K} \longrightarrow \mathcal{P} \times \mathcal{C}$ *is invertible (one-to-one) even though* $\mathcal{G}_P$ *is never onto, then we will write* $\mathcal{G}_P^{-1} : \mathcal{P} \times \mathcal{C} \longrightarrow \mathcal{K}$ *and say* $\mathcal{G}_P^{-1}$ *exists.*

An immediate question arises about the size of plaintext collection P. As we will elaborate later, we seek a polynomial number of plaintexts to have a meaningful attack. For now, assume that there is such a set of plaintext P with a polynomial number, $O(n_b^k)$, of elements so that $\mathcal{G}$ is invertible over this set. Then the boolean circuit, which implements the corresponding ciphertexts in set P, obviously has polynomial gate complexity which we will describe it in detail. Then the question is about the gate complexity of inverse $\mathcal{G}^{-1}$ function. If the inverse complexity is not "too bad" (this is what we will work on) then the circuit structure of Rijndael is not in the majority of block ciphers. So it does not satisfy RSC. Due to Muller's argument (It is given in Theorem 6.1), we know that virtually all functions have optimal circuits with exponential gate complexity. But if the complexity of inverse circuit is exponential as for the majority of block ciphers, then we may claim family of Boolean functions with exponential gate complexity. In any case, the result is amazing that it's an progress on either computational circuit complexity or on block cipher security.

We now specifically outline our potential plan of attack which may disprove the Rijndael Security claim.

We want to determine the key when we are given polynomial number of plaintext-ciphertext pairs. In definition 3.2.4 we define inverse function $\mathcal{G}^{-1}$ for this purpose. Remember that we are working the problem at the circuit level so we have to design the inverse circuit $\mathcal{G}^{-1}$.

If we don't ignore the knowledge of what all these gates are doing, then our analysis wouldn't be at the circuit level. Looking into the structural details of what Rijndael is processing on input, and making up a circuit as a later step is not what we are looking for. So now we have a big circuit and there should be an inverse of this circuit. Under all these assumptions, we will give a concise summary of our attack plan followed by the comments on each of the steps:

(i) Determine the circuit complexity of $\mathcal{F}_p(k)$ where $\mathcal{F}$ is an instance of the scalable Rijndael family.

(ii) Determine if there exists a poly-sized plaintext collection P which identifies the key value for given ciphertexts corresponding to the plaintexts in P.

(iii) Decide if small circuits has 'not too bad' inverse circuits.

1. What is the circuit complexity of $\mathcal{G}_P$ where P has only one plaintext? As a reason that the function $\mathcal{G}$ works independently on elements of P, if we can answer this question, we might upperbound the circuit complexity of $\mathcal{G}_P$ for any plaintext collection P. We are mainly interested with asymptotic complexity. For this purpose we want to define a family of boolean functions simulating

Rijndael in different scales. Smaller scales would also help us to question our doubts in the next steps.

2. Referred to previous discussions and definitions, is there a set of plaintext P such that $\mathcal{G}_P$ is invertible over this set. And how big is P? What if there is no such a polynomial-sized plaintext collection P? In this step we will also question the relationship between one-way functions and the Rijndael Security Claim.

3. If there is such a collection P, and the boolean circuit of invertible $\mathcal{G}_P$ has polynomial gate complexity, then what would the gate complexity of the inverse circuit be? If we can show that, in general, each small circuit has inverse with not "too bad" gate complexity, then our work would be done. To be clear, we do not need a polynomial bound on the gate complexity of inverse in order to disprove the Rijndael security claim. On the other hand each speculation on inverse circuit size gives us very important information.

Chapter 4

Scalable Rijndael Family

In this section we will elaborate on the first step of our attack plan.

The function $\mathcal{G}_P$ where P has only one element p, evaluates the encryption of message p. We can say $\mathcal{G}_P(k) = \mathcal{F}_p(k) = c$, so the circuit of $\mathcal{G}_P$ is exactly same as the Rijndael implementation circuit. Recall our interest of scalable Rijndael like encryption functions. Based on these encryption functions, $\mathcal{G}_P$ can be defined as in ally Definition 3.2.3. So asymptotic bounds on the circuit complexity of $\mathcal{G}_P$ basicly depend on how we define this family of boolean functions.

Let us parameterize the basic elements of Rijndael. These basic elements are the length of byte, the length of word and lengths of 3 different block sizes along with the GF which has characteristic 2. After we construct appropriate round transformations for these parameterized Rijndael family, we want to give complexity analysis of one

encryption. Table 4.1 shows how we parameterize the Rijndael family. For m=2, we definitely have AES, and let's call **Baby Rijndael** to the function parameterized by m=1.

Rijndael(m)	Rijndael(1)	Rijndael(2)	Rijndael(3)	Rijndael(m)
$GF(2^?)$	2^4	2^8	2^{12}	2^{4m}
length of byte	4	8	12	$4m$
length of word	$4 \cdot 2 = 8$	$8 \cdot 4 = 32$	$12 \cdot 6 = 72$	$4m \cdot 2m = 8m^2$
length of block	$8 \cdot 2$	$32 \cdot 4$	$72 \cdot 6$	$8m^2 \cdot 2m = 16m^3$
length of block	$8 \cdot 2 \cdot 1.5$	$32 \cdot 4 \cdot 1.5$	$72 \cdot 6 \cdot 1.5$	$8m^2 \cdot 2m \cdot 1.5 = 16m^3 \cdot 1.5$
length of block	$8 \cdot 2 \cdot 2$	$32 \cdot 4 \cdot 2$	$72 \cdot 6 \cdot 2$	$8m^2 \cdot 2m \cdot 2 = 16m^3 \cdot 2$
number of rounds	5	10	25	$5 \cdot m$

Table 4.1: Parameterization of scalable Rijndael family

We want reader to be aware that Rijndael(m) has polynomial parameters. In order to consider a sequence of instances, one usually has a fairly dense set, or at least a set of instances parameterized by polynomials which is absolutely matching with our case. Recall our aim of having such scalable family; we are mostly interested in claims on family of functions rather than only one example. Also this parameterization helps us to have asymptotic bounds on our explicit problems. In following discussion we give the design of each transformation in round function of Rijndael(m). For simplicity let us take the first block length in the Table 4.1 to answer complexity issues.

1. **SubBytes Transformation**: There are two steps in SubBytes transformation which operate independently on each byte of the state. These steps are first taking multiplicative inverse in the finite field $GF(2^{4m})$ modulo some irreducible polynomial, and then applying affine transformation over GF(2). What we should decide is mainly the irreducible polynomial $p(x) = \sum_{k=0}^{4m} a_k \cdot x^k$ where $a_k \in 0, 1$. For AES we have $p(x) = x^8+x^4+x^3+x+1$, and for Baby Rijndael we can choose $p(x) = x^4 + x + 1$. For large cases where $m > 3$ one can pick any of the irreducible polynomials in $GF(2^{4m})$. Regarding the complexity issues there is no difference what we choose as irreducible polynomial. And for the affine transformation let's have the following equation to update byte value:

$$b'_i = b_i \oplus \bigoplus_{k=0}^{2m-1} b_{(i+2m+k) \pmod{4m}} \oplus c_i$$

for $0 \leq i < 4m$, where b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c. If m=1 then c has the value $\{0011\}$, If m=2 then c has the value $\{01100011\}$, else it has the value m times concatenation of $\{3\}$ (i.e $\{0011\}$). Here and elsewhere, a prime on a variable(e.g., b') indicates that the variable is to be updated with the value on the right. One can check that the definition given above is consistent with the original AES where m is 2.

Complexity Issues: There are various estimations for inversion in $GF(2^k)$, most of them are very efficient but under some circumstances. We are not looking for the optimal solution. The trivial way to take multiplicative inverse

is direct inversion, which is simply equivalent to solving $k \times k$ system of linear equations. We can figure out the problem as this:

We are seeking multiplicative inverse of $A \in GF(2^k)$ where we are given irreducible polynomial $p(x)$. Then we know that $A \cdot B = 1 mod p(x)$ for an element B of $GF(2^k)$. One can extract a system of $k \times k$ linear equations whose solution gives us the polynomial B. Gaussian eliminiation is the way to solve this linear equations system which costs complexity of $O(k^3)$.

Regarding to the affine transformation, without any treatment the obvious complexity is $O(k^2)$ which is fine. So totally one s-box substitution has $O(m^3)$ complexity where the length of byte is $4m$(i.e. transformation is held on Rijndael(m)). The block length is $16m^3$, so we have $4m^2$ number of bytes, then SubBytes transformation costs $O(m^5)$ on the overall. That trivial complexity estimation is better than quadratic since we consider the block length.

2. **ShiftRows Transformation:** Recall that ShiftRows step is a byte transposition that cyclically shifts the row of the state over different offsets. For Rijndael(m), there are $2m$ number of rows which totally constructs each word of the block. So row number i from 0 to $2m - 1$ is shifted over C_i bytes, and we can simply say that $C_i = i$. These offsets values are told for the first block length specified in the Table 4.1, for different block lengths we can design them without disturbing the consistency with AES.

Complexity Issues: Loosely speaking the complexity is $O(m^3)$ which is linear with block length and so better than the complexity of SubBytes transformation.

3. **MixColumns Transformation:** Recall that MixColumns step is a permutation operating on the state column by column. Regarding the parameters of Rijndael(m), the columns of the state are considered as polynomials over $GF(2^{4m})$ and multiplied modula some poynomial $q(x)$ having degree $2m$ with a fixed polynomial $c(x)$. So what we should choose is those $q(x)$ and $c(x)$. Recall that for AES $q(x) = x^4 + 1$ and $c(x) = 03 \cdot x^3 + 01 \cdot x^2 + 01 \cdot x + 02$. For Baby Rijndael let's say $q(x) = x^2 + 1$ and $c(x) = 01 \cdot x + 02$. And one can choose any coprime two polynomials $q(x)$ and $c(x)$ to satisfy MixColumns step for larger scales of Rijndael(m). In the real spesification of AES, there are some criteria of choosing these polynomials to have security resistance against known cryptanalysis. But for our case, we really don't care how one chooses $q(x)$ and $c(x)$.

   Complexity Issues: The modular multiplication with a fixed polynomial can be written as a matrix multiplication. So to determine each byte of the column we apply $2m$ multiplication over $GF(2^{4m})$ by using the irreducible polynomial of SubBytes transformation. Totally it costs $2m \cdot (4m)^2 = 32 \cdot m^3$ to determine one byte of the column. There are $4 \cdot m^2$ number of bytes which means MixColumns transformation costs $O(m^5)$.

4. **Key Addition and Complexity of Rijndael(m):** Key addition layer costs linear in terms of block length. Therefore round function of Rijndael(m) has $O(m^5) + O(m^5)$ complexity. There are $5 \cdot m$ number of rounds which makes the complexity $O(m^6)$, recall that block length was $16 \cdot m^3$, so one can say that Rijndael(m) including AES has overall quadratic circuit complexity.

Chapter 5

Poly-sized Plaintext-Ciphertext Collections

In this section we will elaborate on the second step of our attack plan.

Recall specified P (collection of plaintext), and function $\mathcal{G}$ over this set. If for all k_1 and k_2 there exists at least one $p \in P$ such that $\mathcal{F}_p(k_1) \neq \mathcal{F}_p(k_2)$, then function $\mathcal{G}$ over the set P is invertible. That means, if one gives ciphertexts corresponding to the plaintexts in P, there is exactly one key value which generates all these plaintext-ciphertext pairs. Or in other terms, this plaintext collection P and its corresponding ciphertext collection determine the key.

However, it is possible that there are two key values for which the entire encryption is the same. That implies even non-existence of an exponential sized plaintext

collection which determines the key. In contrast with that, regarding our plan of attack, we are looking for the inverse function $\mathcal{G}_P^{-1}$ where the size of P is polynomial in the length of the plaintext. Because of this potential difficulty we will explore variations of the above problem so that we will be able to continue with our plan of attack. The variations of the problem led us to a theorem of Luby and Impagliazzo. We will also try to say that one-way functions are essential for secure block cipher families in a more concrete way. To obtain a precise meaning of "poly-sized plaintext collection", we are looking at such collections for the whole Rijndael Cipher Family and not only for the individual instances such as the AES Cipher.

It is quite reasonable to believe that there exists even a constant-sized plaintext collection, which determines the key. Recall that, an ideal encryption algorithm has the following assumption: as long as you don't know the underlying key, the input-output behavior of a block cipher closely resembles that of a random permutation. Under this assumption, fixed plaintext encryption of different key values should behave as a random function. Therefore, basically, there is a high probability that there exist two keys encrypting the same way because of the birthday paradox. However, increasing the number of plaintexts in the collection ensures that we could determine the key from the given plaintext-ciphertext pairs. This basic observation leads us to perform small experiments with Baby Rijndael.

The parameters for Baby Rijndael are given in Chapter 4. The block length and

the key length are 16, and the S-boxes are operating on 4 bits. Our example consists of encrypting the message 0^{16} where we say $\mathcal{G}_P(k) = \mathcal{F}_{0^{16}}(k)$. And the result is not surprising; the $\mathcal{G}_P(k)$ function was a permutation, and thus totally invertible.

Despite having this example, it is better to be safe by searching for a mathematical proof for the existence of such $\mathcal{G}_P(k)$ function. However, it seems that it is infeasible to show the existence explicitly. On the other hand, only two key values having the same encryption ends all our discussion but doesn't harm the block cipher security. Because of these 2 reasons (i.e infeasibility and same encryption argument), we would prefer to be more flexible and discuss some variations of the above problem.

In constructing the boolean circuit of a function, one is allowed to use the "don't care" label which is denoted by 'd' in addition to output values '0' and '1'. Specifically, if the output of an input is 'd', this means that the circuit could be constructed such that it evaluates either '0' or '1' for this input, whereas it evaluates correct outputs for any other given input.

Consider that we have some plaintext collection P such that A denotes the image of the function $\mathcal{G}_P()$. Then for each element $E \in A$ (a $|P|$-tuple of ciphertexts), there exists a $k \in \mathcal{K}$ such that $\mathcal{G}_P(k) = E$. As long as the size of A is not small, it is possible to construct a circuit which determines the correct key with high $(|A|/|\mathcal{K}|)$ probability by the use of the 'don't care' trick. This construction is possible if we specify exactly one of the $k \in \mathcal{K}$ satisfying $\mathcal{G}_P(k) = E$ for each $E \in A$. Moreover, for

any other tuple of ciphertext not in the image set A, the constructed circuit outputs any key value by the use of 'don't care'.

'Don't care' has also some other uses for our plan of attack which we prefer to explain in the next chapter. Simply put, even if we have some very big inverse circuit determining the key, whatever the ciphertext collection is, we could construct smaller circuits by the use of 'don't care' idea such that this smaller circuit evaluates the correct key most of the time. In fact, we gave up trying to find the correct key on some occasions with the intention to make the circuit smaller.

In both of the above approaches we understood that $\mathcal{G}_P()$ may not be invertible, but it should have a 'not small' image set. With such flexibility, we want to show that there should be some polynomial-sized plaintext collection P where the image set A of $\mathcal{G}_P()$ is big enough. This is a basic observation allows us to decrease our expectations without harming our plan of attack.

Consider an instance (any cipher) of the Rijndael block cipher family. Let n_b be the block length. Then call the following procedure with parameters n_b and $\mathcal{K}$ (the set of all possible keys).

beginProcedure(m, K_0)**:**

1. if there exists a P so that $\mathcal{G}_P$ is invertible on K_0 where the size of plaintext-

collection P is m^2. (That means for any $k_1, k_2 \in K_0$, $\mathcal{G}_P(k_1) \neq \mathcal{G}_P(k_2)$) return(P, K_0); else continue with following items.

2. Thus, we know that for any plaintext-collection P of size m^2 there exists k_1 and k_2 such that $\mathcal{G}_P(k_1) = \mathcal{G}_P(k_2)$.

3. There are $\binom{2^m}{m^2}$ possible choices for the m^2-tuple P. Each gives a pair (k_1, k_2) out of $\binom{2^m}{2}$ different pairs. Then by the pigeonhole principle there exists a key pair (k_1, k_2) for which at least $\binom{2^m}{m^2}/\binom{2^m}{2}$of these plaintext collections are encrypted identically. Let Y denote this full set of plaintexts appearing in any one of these m^2-tuples for this fixed key pair (k_1, k_2).

4. Then $\binom{|Y|}{m^2} \geq \binom{2^m}{m^2}/\binom{2^m}{2}$

5. So $|Y| \geq \frac{2^m}{2^{2m/m^2}} = 2^{m-\frac{2}{m}}$ which means for this k_1 and k_2 there exists a plaintext-collection Y with $|Y| \geq 2^{m-\frac{2}{m}}$ such that $\mathcal{G}_Y(k_1) = \mathcal{G}_Y(k_2)$.

6. $K_1 = K_0 - \{k_1, k_2\}$

7. return(Procedure(m, K_1));

endProcedure

After starting with Procedure(n_b, K_0) where $K_0 = \mathcal{K}$ is the set of all possible keys, it will return a set K_i and if it is not empty then there would be a plaintext-collection P of size n_b^2 such that $\mathcal{G}_P$ is invertible on K_i. (That means for any $k_1, k_2 \in K_i$,

$\mathcal{G}_P(k_1) \neq \mathcal{G}_P(k_2)$) And for all of the possible keys not in the set K_i we could get pairs of keys giving the same ciphertext for virtually all plaintexts (i.e. same encryption).

Let's speculate on the size of K_i and its complement. If the latter has an exponential size, then the cipher exhibits a weakness. In fact, there would be exponentially many key pairs giving almost the same encryption. If K_i has exponential size, then we would be satisfied since we had been looking for such a poly-sized plaintext collection which gives us an opportunity to determine the key most of the time. On the other hand, one of them should be exponential, since their union set is set of all possible keys which has exponential size

We remark that although the above procedure uses very little about the existence of a poly-sized plaintext collection which determines the key most of the time, non-existence of a restricted version of P (i.e. the size of P is exactly square of the block length) shows a weakness in the cipher. We believe that more careful analysis on encryption of poly-sized fixed plaintext collection leads us to stronger conclusions. On the other hand, the above weakness is enough to discuss the relationship between the security of block cipher family and the existence of one-way functions.

The state of the art regarding the relationship between block ciphers and one-way functions is currently as follows: Since any block cipher is one particular instance of an encryption scheme, it isn't directly possible to relate it to one-way functions. In [ImLu89] Impagliazzo and Luby first showed that any secure private key encryption

algorithm can be used to construct a pseudo-random generator. Moreover, the construction of a pseudo-random generator is equivalent to the existence of a one-way function.

However, the Rijndael proposal led us to define a scalable block cipher family. So we are now able to relate the security of the Rijndael cipher family directly to the existence of one-way functions. The impact of this shouldn't be underestimated. Because claiming the RSC in terms of Rijndael's authors is now equivalent to claiming a well-defined and implementable one-way function family. The following theorem gives a simple and basic sketch of the relationship. This theorem is applicable to any block cipher family. R_n denotes the Rijndael-like cipher parameterized by n.

Theorem 5.1 *Suppose there exists an integer n_0 such that R_n is K-secure and hermetic for all $n \geq n_0$. Then there exist plaintext collections $P_n(n > 0)$ of polynomial size such that for all probabilistic poly-time algorithms A and for any positive polynomial $p(n)$ there exists n_1 such that following holds for all $n > n_1$:*

$$Pr_U[A(\mathcal{G}_P(U)) \in \mathcal{G}_P^{-1}(\mathcal{G}_P(U))] \leq \frac{1}{p(n)}$$

In other words, if the Rijndael family function is K-secure and hermetic for all sufficiently large n, then one may find a one-way family function $\mathcal{G}_P$ which is constructed by use of parameterized-Rijndael encryption.

Remark: That would definitely imply $P \neq NP$. The above definition for one-way function family is taken from Goldreich [Go01].

Proof: Our proof is by contradiction. Assume n_0 exists. Yet assume that the conclusion is false. That is, assume for all poly-sized plaintext collections P, there is some algorithm A and some positive polynomial $p(n)$ such that for all n, there is some $m > n$ with:

$$Pr_U[A(\mathcal{G}_P(U)) \in \mathcal{G}_P^{-1}(\mathcal{G}_P(U))] > \frac{1}{p(m)} \tag{5.1}$$

In other words, if we are given the output of function $\mathcal{G}_P$ for any polynomial-sized plaintext collection, there exists a probabilistic polynomial-time algorithm that would give us one of the pre-images of this output with probability at least $1/p(m)$.

By our assumption, with $n = m_i$ and $m_{i+1} > m_i$ being chosen to satisfy 5.1, we obtain a sequence of m_i going to infinity for which any plaintext-collection P of size m_i^2, for example, there exists a probabilistic polynomial-time algorithm A and a polynomial $p()$ for which inequality 5.1 holds. (**)

For each $m_i, i \geq 1$, trace the procedure we gave early in this section until it returns.

After starting with Procedure(m_i, K_0) where K_0 is set of all possible keys, for each m_i we get a set K_i and if it is not empty then there would be a plaintext-collection P_{m_i} of size m_i^2 such that $\mathcal{G}_{P_{m_i}}$ is invertible on K_i (That means for any $k_1, k_2 \in K_i$, $\mathcal{G}_{P_{m_i}}(k_1) \neq \mathcal{G}_{P_{m_i}}(k_2)$) and for all of the possible keys not in the set K_i we could get pairs of keys giving the same ciphertext for virtually all plaintexts (i.e. same

encryption).

If there is a subseries of K_i going to infinity such that for any polynomial $t(n)$, $|K_i| \geq (2^{m_i} - \frac{2^{m_i}}{t(m_i)})$ eventually holds and $\mathcal{G}_{P_{m_i}}$ is invertible on K_i, then for all n_0 there exists $m_i > n_0$ such that R_{m_i} is not K-secure. This holds because from (**) there is a probabilistic polynomial-time algorithm A which determines one of the pre-keys with probability greater than $1/p(m_i)$. Now choose $t(n) = 2p(n)$. We know that there is a set of K_i on which $\mathcal{G}_{P_{m_i}}$ is invertible and $|K_i| \geq (2^{m_i} - \frac{2^{m_i}}{2 \cdot p(m_i)})$, so the number of keys not in the set K_i is smaller than $\frac{2^{m_i}}{2 \cdot p(m_i)}$. Therefore, with at least $\frac{1}{2 \cdot p(m_i)}$ probability, algorithm A strictly determines the key if it determines one of the pre-keys.

So there exists a polynomial $t(n)$ for which $|K_i| < (2^{m_i} - \frac{2^{m_i}}{t(m_i)})$ for sufficiently large values of m_i. For such m_i, the number of pairs of keys with the same encryption is greater than $(2^{m_i} - (2^{m_i} - \frac{2^{m_i}}{t(m_i)}))/2 = \frac{2^{m_i - 1}}{t(m_i)}$. That means with very high probability a randomly chosen key has an equivalent one giving almost the same encryption. That definitely contradicts the hermetic property of R_{m_i} (according to our previous discussions in Section 2.3) and there would be an $m_i > n_0$ whatever the value of n_0 is. ♣

In the proof, we assumed the claim of Daemen and Rijmen that the existence of key-finding algoritm faster than exhaustive search is a violation of K-security. But to be careful, perhaps we should allow for "generic" key-recovery attack algorithms. By this, we mean a probabilistic polynomial time key-recovery attack which applies to a

majority of block ciphers of a given length.

It is important to note that the above analysis is not particular to Rijndael but gives information when applied to infinite sequence $\{B_n\}$ of block ciphers. For example, suppose there exists an infinite sequence $\{B_n\}$ of block ciphers which are all K-secure and hermetic. Note that we do not assume any uniform description for these ciphers. Applied to this sequence, the theorem gives one of three outcomes:

1. For any polynomial $f(n)$ the key length of B_n is eventually greater than $f(n)$.

2. One-way functions exist.

3. There exist infinitely many key lengths suspectible to a generic key-recovery attack algorithm.

Chapter 6

Inverting Circuits

In this chapter, we want to elaborate on the third step of our attack plan.

Recall specified P (collection of plaintext), and function $\mathcal{G}_P$ over this set. If for all k_1 and k_2 there exists at least one $p \in P$ such that $\mathcal{F}_p(k_1) \neq \mathcal{F}_p(k_2)$, then the function $\mathcal{G}_P$ defined over the set P is invertible. That means, if one gives ciphertexts corresponding to the plaintexts in P, there is exactly one key value which generates all these plaintext-ciphertext pairs. In other words, this plaintext collection P determines the key.

In the third step of our attack plan, we assume that $\mathcal{G}_P^{-1}$ exists for some set P of plaintexts where $\mid P \mid = poly(n_b)$. We seek to investigate the circuit complexity of $\mathcal{G}_P^{-1}$. Finding any meaningful upper or lower bound on this special circuit complexity would be of interest in computational complexity and in the security of block ciphers.

On the other hand, regarding the discussions in the previous chapter about our plans' success, it is possible to be flexible by seeking an inverse circuit which determines the key most of the time. This flexibility has two aspects. One aspect covers the case in which there is no poly-sized P. Then we are allowed to modify some $\mathcal{G}_P$ to make it invertible but inverse of this modified one should determine the key most of the time. The other aspect gives us opportunity to modify the inverse circuit although $\mathcal{G}_P$ is invertible since it would be satisfactory to determine the key with high probability as long as the modified inverse circuit has small complexity.

Although we have such flexibilities to break the Rijndael security claim (RSC), we want to concentrate on the pure and rawest question about the gate complexity of inverse circuits. So, in this chapter we will deal with mostly the inverse circuit of any invertible $\mathcal{G}_P$.

To make a clear organization of thoughts on this specific problem, first we prefer to mention our motivations. Then we want to interpret existing theory on complexity of circuits along with known results on inverse circuits. This discussion will be followed by some examples on inverting small permutation circuits. We will try to give some methodology for future research.

The problem in our research outline is to find the upper bound for circuit complexity of $\mathcal{G}_P^{-1}$. Existence of any circuit with $poly(n_b) \cdot 2^{n_b(1-c)}$ gates, where n_b is block length and c is constant (such as $c = 0.0001$), constitutes an exponential improve-

ment on the circuit complexity. Therefore, this circuit allows us to apply an attack on the Rijndael which is better than exhaustive search. Although drawing bounds on explicit problems in computational complexity, as we will mention later, is a very hard task, we know that even the hardest NP-complete problems have exponential improvements at the algorithmic level. Recent papers on the SAT problem give examples of exponential improvements ([DaHiWo04], [PaPu98], [DaWo04]). Thus there seems to be a good chance for finding an exponential improvement, which is clearly contrary to the Rijndael security claim; this is one of the main motivations of this outlined research.

Our second motivation is based on the comparison of circuit complexity and its inverse complexity. It is quite reasonable to believe that small circuits do not have 'very bad' inverse circuits. And as we show in Chapter 4, the Rijndael block cipher family has at most quadratic gate complexity.

Computational complexity tries to give lower bounds on the inherent computational difficulty of certain problems. In his famous paper [Sh49], Shannon suggested the size of Boolean circuits as a measure of computational difficulty. Among computational models, the circuit model has a simple definition so it is more appropriate for combinatorial analysis, but unfortunately, despite much intensive work on circuit complexity only very weak lower bounds on circuit size are known, and only for very few decision problems.

This discipline is mostly concerned with explicit problems. But by using counting arguments, it is not hard to establish bounds on nonexplicit problems. Probabilistic methods are mostly used for finding average bounds on general version of decision problems. According to bounds probabilistically found by Muller [Mu56] based on Shannon's argument [Sh49], the following is dramatically true:

Theorem 6.1 (Muller, 1956) *Almost every Boolean function of n variables requires circuits of size* $\Omega(2^n/n)$.

In other words, there exist a $c > 0$ such that if f is a randomly chosen boolean function of n variables, then with probability approaching 1 as n goes to infinity, any circuit computing f has at least $c(2^n/n)$ gates.

Despite the importance of lower bounds on the circuit complexity of explicit problems, the best bounds known are only linear. Therefore, although we know that many exponential-sized boolean functions exist, nobody has been able to come up with a natural family of Boolean functions that requires more than a linear number of gates to compute. To state the problem more clearly, we can say there is no specific optimum circuit whose behavior isn't random and whose gate complexity is more than linear. Blum [Bl84] proved a $3n - o(n)$ lower bound for such that, we could construct a family of Boolean functions with this lower bound.

In this part of our attack plan, we are questioning computational asymmetry between circuit and its inverse circuit. In terms of circuit complexity, unconditional

secrecy would be possible with any invertible function with great computational asymmetry.

Let T_n be the set of permutations on $\{0,1\}^n$ (i.e., the set of invertible functions from n bits to n bits) Basically, there has not been much progress on finding one-way functions in T_n, i.e., invertible functions with great computational asymmetry. Finding such a function would seem to be a great step on the way to a theory of cryptography for computational security.

Alain Hiltgen holds the current world record for computational asymmetry for constructive functions in T_n. He could, for every n, construct a function whose inverse requires twice as many gates as the function itself! [Hi93] Moreover, Massey has the following proposition:

Proposition 6.1 (Massey, 1996) *For all $n \geq 6$, virtually all functions in T_n have gate complexity that differs by a factor of less than 2.5 from the gate complexity of their inverse function.*

We refer the reader to [Ma96] for a proof of above proposition as well as more information on gate complexity. At first look, it seems that this proposition would solve all our problems. However, there is a simple trick on the above proposition and this trick do not apply for our case and for functions with small circuit complexity. Simply put, for all $n \geq 6$ virtually all functions have circuit complexity of $\Theta(2^n/n)$, stronger than Theorem 6.1 which results that virtually all functions have the gate

complexity that differs by a constant factor from the gate complexity of their inverse function.

Alain Hiltgen and Jurg Ganz also found the first ever computationally asymmetric permutation for the gate complexity measure; that is in T_4. According to their exhaustive search on functions that are more difficult to invert than to compute, T_1, T_2 and T_3 are computationally symmetric. Here is the asymmetric permutation in T_4 (Note that $\oplus$ represents 'xor' and $\bullet$ represents 'and' operation):

$$y_1 = x_1 \oplus x_2, y_2 = x_2 \oplus [(x_1 \oplus x_2) \bullet x_4], y_3 = x_3 \oplus [(x_1 \oplus x_2) \bullet x_4], y_4 = x_4$$

The gate complexity of the above function is 5, and its inverse, which has gate complexity of 6, is as follows:

$$x_1 = [(y_1 \oplus y_3) \oplus y_2] \bullet y_4 \oplus (y_1 \oplus y_3), x_2 = [(y_1 \oplus y_3) \oplus y_2] \bullet y_4 \oplus y_2$$

$$x_3 = [(y_1 \oplus y_3) \oplus y_2] \bullet y_4 \oplus y_3, x_4 = y_4$$

Now, we want to talk on the methodology for solving the problem of whether small circuits with sufficient diffusion has 'not big' inverse circuits. Circuit level analysis is not an easy job, best explicit constructions on 'big circuits' (recall we have $3n - o(n)$ lower bound on circuit complexity) and on functions of great computational asymmetry (recall we have only Alain's function) exhibits the hardness of the problem.

We want to deal with functions having diffusion. Diffusion is the main requirement for a block cipher proposal. That's the reason why we want to restrict our analysis

on functions in T_n which at least suffice this requirement. Basically, if a function has diffusion, then each output bit is dependent on each input bit as well as every input bit influences every output bit. Considering a function with sufficient diffusion leads us to the following proposition on the gate complexity of this function:

Proposition 6.2 *Any function in T_n with sufficient diffusion requires at least linear gate complexity.*

Proof: The proof simply relies on the fact that each output bit should be an output of a gate, since we have diffusion. However, different output bits have different end gates, because the function is in T_n, so two output bits can not be same all the time.♣

To analyze the problem of inverting circuits, one categorizes the circuit sizes of functions and tries to ask one by one. So at first, it is reasonable to ask the gate complexity of inverse circuits of functions having linear complexity. On the other hand, it is not possible to invert functions independently for each different gate complexity classes. Mainly, we believe that at some point getting an answer for functions having linear gate complexity becomes equivalent to getting an answer for functions having quadratic or even greater polynomial gate complexity. This belief can be observed by the following two claims.

Proposition 6.3 *Given any invertible circuit $C : \{0,1\}^{\log m} \to \{0,1\}^{\log m}$ with $O((\log m)^2)$ gates, there exists an invertible circuit $C' : \{0,1\}^m \to \{0,1\}^m$ with $O(m)$ gates and*

sufficient diffusion which contains C and whose inverse gate complexity is somehow related to inverse gate complexity of C.

Proof: Take an invertible circuit $D : \{0,1\}^m \rightarrow \{0,1\}^m$ with $O(m)$ gates and sufficient diffusion. In addition its inverse D^{-1} must have gate complexity of $O(m)$. On input x, the circuit C' first applies circuit C on its last $\log m$ bits, then applies circuit D to all its bits. Therefore, $C'(x_1, x_2) = D(x_1, C(x_2))$ where $|x_1| = m - \log m$ and $|x_2| = \log m$. A possible inverse circuit for C' is the one first applies inverse D^{-1} and then C^{-1}. So the optimal inverse circuit of C' has complexity of less than $O(m) + Comp(C^{-1})$ where $Comp(C^{-1})$ denotes the optimal gate complexity of C^{-1} (1). On the other hand, to invert C, one can first pad more information to input so that he has a string of length m. Then applies D to it, now he has an possible output for circuit C'. If he applies inverse circuit of C', the last $\log m$ bits of resulting string denotes the inverse for C. So the optimal inverse circuit of C has complexity of less than $O(m) + Comp(C'^{-1})$ where $Comp(C'^{-1})$ denotes the optimal gate complexity of C'^{-1} (2). By combining the results (1) and (2), the inverse gate complexity of C' changes linear with the inverse gate complexity of C. ♣

We could modify the above claim to get a similar claim, but its implications would be different than the above one. The proof of the following claim is the same as the previous one.

Proposition 6.4 *Given any invertible circuit* $C : \{0,1\}^m \rightarrow \{0,1\}^m$ *with* $O(m^k)$

gates, there exists an invertible circuit $C' : \{0,1\}^{m^k} \rightarrow \{0,1\}^{m^k}$ *with* $O(m^k)$ *gates and sufficient diffusion which contains* C *and whose inverse gate complexity is somehow related to inverse gate complexity of* C*.*

In regards to both observations, although there exists an invertible C' related to the C, it doesn't harm the fact that functions with linear complexities may have 'not big' inverse circuits. Even if C has a 'very bad' inverse circuit, C' still has the inverse circuit which doesn't look like the random case and its inverse is exponentially better than the worst case. However, these simple observations point out the fact that it is not an easy job to categorize functions with different gate complexities and to solve them independent from each other.

We had been trying to find 'one-way functions' in terms of gate complexity. However, the nature of the problem and existing weak results on circuit complexity demonstrates that this has been an open problem since Massey first mentioned in 1996. Even we have limited knowledge on one-way functions in uniform model of computation. But recall our main quest on RSC, so we are looking for the inverse of the $\mathcal{G}_P$ function which is not any circuit and exhibits lots of specialties. After now, we will concentrate on inverting $\mathcal{G}_P$, and on seeking the gate complexity of $\mathcal{G}_P^{-1}$.

Naively considering the circuit complexity of $\mathcal{G}_P^{-1}$, one immediately recalls Muller's theorem (Theorem 6.1). If $\mathcal{G}_P^{-1}$ behaves like a random function, any circuit representing it requires an exponential number of gates. Although the input size of $\mathcal{G}_P^{-1}$ is $m \cdot n_b$,

we can say Muller's argument works for m independent group of n_b bits. That observation is important, otherwise average circuit complexity of $\mathcal{G}_P^{-1}$ is $\Omega(2^{m \cdot n_b}/m \cdot n_b)$ which is bad news for us. Neither $\mathcal{G}_P^{-1}$ nor $\mathcal{G}_P$ is a random function working on $m \cdot n_b$. Each output bit of $\mathcal{G}_P$ is absolutely determined by only n_b bits because it is an encryption function. Similarly any output (length of n_k) bit of $\mathcal{G}_P^{-1}$ is evaluated by only n_b bits. We need m independent group of n_b bits to determine the output uniquely.

So random $\mathcal{G}_P^{-1}$ function has $m \cdot n_b \cdot \Omega(2^{n_b}/n_b) = m \cdot \Omega(2^{n_b})$ gate complexity which is at least $\Omega(2^{n_b})$. However neither $\mathcal{G}_P$ nor $\mathcal{G}_P^{-1}$ is random. $\mathcal{G}_P$ has simple structure with a polynomial number of gates as mentioned in Chapter 4. So $\mathcal{G}_P^{-1}$ is not something completely out of control.

Let's note what security claim (Rijndael cipher is K-secure and hermetic) tells about the circuit complexity of $\mathcal{G}_P^{-1}$. Informally speaking, any attack on a K-secure block cipher has the same expected work on virtually all block ciphers. Recall the number of all block ciphers in Section 2.2, that is $(2^{n_b}!)^{2^{n_k}}$. The number of block ciphers having invertible $\mathcal{G}_P$ for plaintext collection P with polynomial number of elements is obviously smaller than $(2^{n_b}!)^{2^{n_k}}$. Among these, virtually all inverse $\mathcal{G}_P^{-1}$ function has exponential circuit complexity ($\Omega(2^{n_b})$) according to Muller's argument. So it is quite reasonable to believe that being hermetic implies the fact that this inverse $\mathcal{G}_P^{-1}$ function has $\Omega(2^{n_b})$ gate complexity.

In this chapter, we first mentioned the theoretical results on inverting any circuit,

we tried to point its hardness by talking about the methodology to solve such a problem. However, we simply saw that we are more comfortable to break RSC, since the Rijndael block cipher family is not chosen random and it's small enough.

Chapter 7

Conclusions

Our main motivation for starting this research was to introduce a fresh strategy to attack the Rijndael security claim (RSC). We were skeptical of RSC at the theoretical level. Aiming to break the very bold Rijndael security claim, we gave a formal understanding of RSC juxtaposing it with rigorous security treatments.

The main steps of our attack strategy are summarized as follows:

(i) Determine the circuit complexity of the encryption function $\mathcal{F}$ where $\mathcal{F}$ is an instance of the scalable Rijndael family.

(ii) Determine if there exists a poly-sized plaintext collection P which identifies the key value for given ciphertexts corresponding to the plaintexts in P.

(iii) Decide if small circuits have 'not too bad' inverse circuits.

To answer the questions above, we constructed a scalable family of ciphers which

behave very much like the original Rijndael. This construction gave us the opportunity to generate a candidate one-way function family. However, this endeavor totally depended on the second step of our attack strategy. Basically, we argued the existence of poly-sized plaintext collection by relating RSC to its non-existence. Therefore, we proved that if there is no such poly-sized collection, then RSC fails. On the other hand its existence directs us to the generation of a candidate one-way function family.

We noted that the Rijndael proposal allows a reduction to one-way functions; it is rare to have such a defense of security in the private-key theatre.

Regarding the 3^{rd} step of attack strategy we explored the theoretical results on inverting any circuit. We introduced some propositions which address the hardness of the question.

7.1 Future Research

We believe that in the non-uniform model of boolean circuit complexity, an adversary of a cipher would determine the key by any exponential amount of precomputation so long as the results of this computation can be encoded in polynomial-sized circuits. Specifically, Rijndael-like iterated block ciphers have better than worst-case inverse circuits which somehow related to key-recovery. In particular, invertible functions represented by circuits with very small numbers of gates have better than worst case implementations for their inverses. However, the research in cryptography which

accepts gate complexity as a measure of security is still at an early stage. Possible directions to future research on this topic are listed below:

1. It seems a natural requirement for block cipher proposals with such bold security claims (i.e. RSC) to include scalability of the proposal. Rijndael proposal is the first well-known block cipher that supports scalability. Computational complexity techniques are now available to attack such bold security claims.

2. We proved existence of poly-sized plaintext collection which determines the key implicitly by saying that its non-existence implies the failure of RSC. However, we believe that a constant-sized plaintext collection determines key. Finding such poly-sized plaintext collection explicitly would be an interesting result, and this progress would explore some information on block cipher security.

3. There should be an inverse circuit somehow related to key-recovery where the encryption circuit has small number of gates. Therefore the following question in its purest state is helpful to answer: "Do invertible functions represented by circuits with very small numbers of gates have better than worst case implementations for their inverses?"

4. Rijndael-like iterated block ciphers are not like any small invertible circuit, their circuit level description contains much more information. There may be some promising properties that allow inverting a circuit by precomputing exponential

amount of time so long as the results of this computation can be encoded in polynomial-sized circuits.

Bibliography

[Be98] Mihir Bellare, "Practice-Oriented Provable-Security," Lecture Notes in Computer Science Vol. 1936, Springer Verlag, 1998.

[BeRo93] Mihir Bellare and Phillip Rogaway, "Entity Authentication and key Distribution," Advances in Cryptology-Crypto 93 Proceedings, Lecture Notes in Computer Science Vol. 773, Springer Verlag, 1993

[BeRo94] M.Bellare, P. Rogaway and J. Kilian, "The Security of block cipher chaining," Advances in Cryptology-Crypto 94 Proceedings, Lecture Notes in Computer Science, Springer Verlag, 1994

[BeRo97] M. Bellare, P. Rogaway, A. Desai and E. Jokipii, "A Concrete Security Treatment Of Symmetric Encryption," Proceedings Of the 38th Symposium on Foundations Of Computer Science, IEEE, 1997.

[BiSh91] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," Journal of Cryptology, Vol. 4, No. 1, 1991, pp. 3-72

[Bl84] N. Blum, A Boolean function requiring 3n network size, Theoret. Comput. Sci. 28 pp. 337-345, 1984.

[Br83] Brassard G., "Relativized Cryptography", IEEE Trans. Inform. Theory 29, pp. 877-894, 1983.

[BoLa87] R. B. Boppana and J. C. Lagarias, "One-Way Functions and Circuit Complexity", Information and Computation vol. 74, pp. 226-240, 1987.

[Da95] J. Daemen, "Cipher and hash function design strategies based on linear and differential cryptanalysis," Doctoral Dissertation, March 1995, K.U.Leuven.

[DaHiWo04] Evgeny Dantsin, Edward A. Hirsch, Alexander Wolpert, Algorithms for SAT Based on Search in Hamming Balls. STACS 2004: 141-151

[DaWo04] Evgeny Dantsin, Alexander Wolpert, "Derandomization of Schuler's Algorithm for SAT," Electronic Colloquium on Computational Complexity (ECCC)(017): (2004)

[DiHe76] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. Info. Theory, Vol. IT-22, No. 6, November 1976, pp. 644-654

[DR99] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. In AES Round 2 Technical Evaluation, CD-2 Documentation. NIST, March 1999. See http://www.nist.gov/aes

[DR01] Joan Daemen and Vincent Rijmen. The Design Of Rijndael:AES- The Advanced Encryption Standard, Springer-Verlag, 2001.

[Go01] Oded Goldreich. Foundations Of Cryptography, Cambridge University Press, 2001

[Gl02] Brian Gladman. A Specification for Rijndael, the AES Algorithm, 2002. See " http://www.techheap.com/cryptography/encryption/spec.v36.pdf"

[GoMi84] S. Goldwasser and S. Micali, "Probabilistic Encryption," J. of Computer and System Sciences, Vol. 28, April 1984, pp. 270-299.

[GoMiGo86] O. Goldreich, S. Micali and S. Goldwasser, "How to construct random functions," Journal of the ACM, Vol 33, No. 4, pp. 792-807, October 1986

[Hi93] Alain Hiltgen, "Constructions of Feebly-One-Way Families Of Permutations," Advances in Cryptology AUSCRYPT'92, Lecture Notes in Computer Science No. 718. New York: Springer, 1993, pp. 422-434.

[ImLu89] R. Impagliazzo and M. Luby, "One-way Functions are Essential for Complexity Based Cryptography," Proc. of the 30th Annu. IEEE Syrup. on Foundations of Computer Science, pages 230-235, 1989.

[Kn95] L.R.Knudsen, "Truncated and higher order differentials," Fast Software Encryption, LNCS 1008, B. Preneel, Ed., Springer-Verlag, 1995, pp. 196-211.

[LeComp] Introduction to Modern Cryptography, Lecture Notes of M. Bellare and P. Rogaway, See "http://www-cse.ucsd.edu/users/mihir/cse207/classnotes.html"

[Le87] Levin L. A., "One-way functions and pseudorandom generators", Combinatorica, proceedings, 17th Annual ACM Symposium on Theory of Computing, pp. 363-365, 1987.

[Lee90] J. Van Leeuwen, Handbook of Theoretical Computer Science, The MIT Press, 1990.

[LNCP] Lecture Notes of Christof Paar, See "http://ece.wpi.edu/ sunar/ee578/chapter5.pdf"

[Ma04a] James L. Massey, "Is a Mathematical Theory of Cryptography Possible?," See "www-rocq.inria.fr/fr/actualites/colloquium/archives/INRIA_ 2004_ Math_ Theory_ Crypto.ppt"

[Ma04b] James L. Massey, "Secrecy and Difficulty," See "http://www.nd.edu/ cam/ima/CourseMateria

[Ma94] M. Matsui, "Linear cryptanalysis method for DES cipher," Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765, T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386-397.

[Ma96] James L. Massey, "The Difficulty with difficulty," EUROCRYPT'96 IACR Distinguished Lecture, See "http://www.iacr.org/publications/dl/massey96/html/massey.html".

[Mu56] D.E. Muller, Complexity in electronic switching circuits, IRE Trans. Electronic Computers 5, pp. 15-19, 1956.

[NIST01] Federal Information Processing Standards Publication (FIPS). Advanced Encryption Standard, FIPS, NIST/2001. See http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[Pa95] Christos H. Papadimitriou, Computational Complexity, Addison-Wesley Inc. 1995.

[Papu98] R. Paturi, P. Pudlak, M. E. Saks, and F. Zane. "An improved exponential-time algorithm for k-SAT," Proc. 39th Symp. Foundations of Computer Science, pp. 628-637. IEEE, 1998

[Se84] Selman A. L., "Remarks about natural self-reducible sets in NP and complexity measures for public key cryptosystems", 1984.

[Sh49] C. E. Shannon, "The synthesis of two-terminal switching circuits", Bell Systems Tech. J. 28(1) pp. 59-98, 1949.

[Sti95] Douglas R. Stinson, Cryptography; Theory and Practice, CRC Press, 1995.