Integral Cryptanalysis

Vebjørn Moen

November 21, 2002

blank page

Abstract

This master thesis looks at cryptanalysis of block ciphers, and specially integral cryptanalysis. We show that Nyberg's generalized Feistel networks[36] are vulnerable to integral cryptanalysis, and we present some attacks on these ciphers. We also show that the generalized Feistel networks with parameters equal to the parameters of the 128-bit key and block version of the AES, are more vulnerable to integral cryptanalysis than the AES.

Keywords: cryptanalysis, block ciphers, integral, generalized Feistel networks, Rijndael, AES, MISTY, KASUMI.

Acknowledgment

First of all I would like to thank my supervisor Lars R. Knudsen for introducing me to the exciting world of cryptology and for always taking time to answer my questions. I would also like to express my gratitude to Pål Berg at Sospita for excellent comments and suggestions to this thesis, and to Håvard Raddum for helpful discussions.

Also thanks to Tor Helleseth and the coding theory and cryptography group at the University of Bergen for financial support for Fast Software Encryption and Eurocrypt, and Sospita for financial support for Norwegian Crypto Seminar.

Thanks to my family, especially Cecilie and Marie Bjørk, for all support and encouragement they have given me.

Vebjørn Moen
Bergen, 26 April 2002

Contents

1 Introduction **4**

2 Introduction to Cryptology **6**
2.1 Cryptography 6
2.1.1 Design Principles 7
2.1.2 Block Ciphers 7
2.2 Cryptanalysis 9
2.2.1 Classification of Attacks 11
2.2.2 Classification of Breaking Ciphers 11
2.2.3 Complexity of Attacks 12

3 Some Attacks on Block Ciphers **13**
3.1 Differential Cryptanalysis 13
3.1.1 Some Definitions 13
3.1.2 Differentials 14
3.1.3 Principle of a General Attack 15
3.1.4 Complexity 16
3.2 Linear Cryptanalysis 16
3.2.1 Introduction 16
3.2.2 Principle of a General Attack 17
3.2.3 Complexity of the Attack 18
3.3 Higher Order Differentials 18
3.4 Truncated Differentials 19
3.5 Interpolation Attack 19
3.5.1 Global and Instance Deduction 20
3.5.2 Key-recovery 20
3.6 Other Attacks 21

4 Resistance Against Differential Attacks **22**
4.1 Some Results 22
4.2 Conclusion Of This Chapter 23

5 Integral Cryptanalysis of Block Ciphers **24**
5.1 Introduction 24
5.2 Higher Order Integrals and Some Notations 26
5.3 Comparison to Other Attacks 27
5.4 The Square Attack 28
5.4.1 The Basic Attack 28

5.4.2 Extension by an Additional Round at the End 30
5.4.3 Extension by an Additional Round at the Beginning . . . 30
5.5 Attacks on Generalized Feistel Networks 32
5.6 Attacks on MISTY 33

6 New Integrals **37**
6.1 Generalized Feistel Networks 37
6.1.1 First Order Integral 37
6.1.2 Second Order Integral 41
6.1.3 Even Higher Order Integrals 43
6.1.4 Implementation of the Attacks 44
6.2 KASUMI 44
6.3 Conclusion 45

A Higher Order Derivatives **49**

B MISTY **50**

C Implementations **54**
C.1 Generalized Feistel Network 54
C.2 The Attack Using a 1. Order Integral 55
C.3 Integral Finder 55

Chapter 1

Introduction

Encryption, or using secret codes for private communication, dates back thousands of years. How to protect communication and information by secret codes is known as cryptography. An example of such encrypted communication is the Roman Emperor Julius Caesar, who encoded messages to his generals to protect the content against the possibility of interception. More recently, Alan Touring led a group of British mathematicians who broke the German code, Enigma, used in World War II for sending instructions to U-boats patrolling the Atlantic Ocean. Governments still depend on secret codes, and much of the work that have been done in this area were not and is not published.

While the first computers were owned by Governments and large corporations and had to be placed in specially built rooms, today's personal computers can fit into a hand. Smaller and cheaper computers and the development of computer networks and the Internet have resulted in computers in almost every home. This has increased the interest and need for secure communication. In recent years, corporations and individuals have started to use encryption to secure their electronic information and communication. Digital cellular telephones, direct satellite television broadcast and electronic commerce over the Internet, all depend on cryptographic measures to protect information. Soon, nearly all electronic communication will be cryptographically secured.

It is only since the mid-seventies that cryptography has been an academic field of research. In 1975, the National Bureau of Standards (NBS) proposed a Data Encryption Standard (DES). What the Bureau published was an IBM design with changes recommended by the National Security Agency (NSA), including a shorter key length, reduced from 64 bits to 56 bits. In 1977, the DES (with a 56-bit key) was issued as a Federal Information Processing Standard (FIPS). Until 1991, when Biham and Shamir published the differential attack[6], no attack faster than exhaustive search was known for the DES. The DES has been and is still used in many systems.

Ever since the DES was published it has been criticized for too small key-size, and in the end of the 1990s the National Institute of Standards and Technology started a new process to select the block cipher that is going to be the standard block cipher for commerce and US Government in the next 2 or 3 decades to come. This standard is called the Advanced Encryption Standard (AES), and much work has been done during the process of develop the new standard. New attacks and design criteria have been found during this process. This was an

international and open competition, and the winner, a proposal from Belgium: Rijndael [10], was selected in October 2000.

Several new attacks on block ciphers have been discovered during the work to find a replacement for the DES, both previous to and during the AES process. This thesis will consider one of these attack. This attack was formerly known as the Square attack by Knudsen[9], then generalized by Knudsen and Wagner[23] and called *integral cryptanalysis*. This is the best known attack on the AES at the present time, and in this thesis it is investigated if the generalized Feistel networks proposed by Kaisa Nyberg[36] have better resistance than the AES against integral cryptanalysis.

The structure of the rest of this thesis is as follows: Chapter 2 gives some definitions concerning cryptology, Chapter 3 looks at some previous attacks on block ciphers. Chapter 4 gives a brief introduction to how a cipher can be made resistant against differential and linear attacks. Chapter 5 provides the basis for the integral attack, and gives some examples of (some reduced variants of) ciphers that have been attacked with integral cryptanalysis. In Chapter 6 some new integrals and attacks are presented, and we draw some conclusions. The main new results from this thesis are new integrals and attacks on the generalized Feistel networks which show that this structure does not give any advantage over the design of the AES.

Chapter 2

Introduction to Cryptology

The main idea in the development of cryptology is to make it possible for two parties, traditionally called Alice and Bob, to have a secure communication over an insecure channel where some enemy, often called Eve, can listen in on the communication. The goal is that Eve should not learn anything about the original message from what she sees on the line. Cryptology is a common descriptor for cryptography and cryptanalysis. Cryptography is where we try to make a message unreadable for anyone other than those who possess the right "key". Cryptanalysis is where we try to read an encrypted message, or to find the secret key, without knowing the key in advance. Words like attack or break a cryptosystem are common for cryptanalysis.

2.1 Cryptography

A cryptosystem is often referred to as a cipher, and the formal definition of a cryptosystem can be stated as:

Definition 2.1.1 *[42] A cryptosystem is a five-tuple* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, where the following conditions are satisfied:*

1. $\mathcal{P}$ *is a finite set of possible plaintexts*
2. $\mathcal{C}$ *is a finite set of possible ciphertexts*
3. $\mathcal{K}$*, the key space, is a finite set of possible keys.*
4. *For each* $K \in \mathcal{K}$*, there is an encryption rule* $e_K \in \mathcal{E}$ *and a corresponding decryption rule* $d_K \in \mathcal{D}$*. Each* $e_K : \mathcal{P} \to \mathcal{C}$ *and* $d_K : \mathcal{C} \to \mathcal{P}$ *are functions such that* $d_K(e_K(x)) = x$ *for every plaintext* $x \in \mathcal{P}$*.*

A symmetric cryptosystem consists of a sender and a receiver who share a secret called a key. When the sender, Alice, wants to send a message to Bob, she encrypts the message, also called the plaintext, using some encryption algorithm and a secret key agreed upon in advance. Alice sends the result, called ciphertext, over an insecure channel to Bob. Bob decrypts the ciphertext with the secret key using the some decryption algorithm. If Eve is listening to the insecure channel, she will only see the ciphertext which to her is gibberish. This

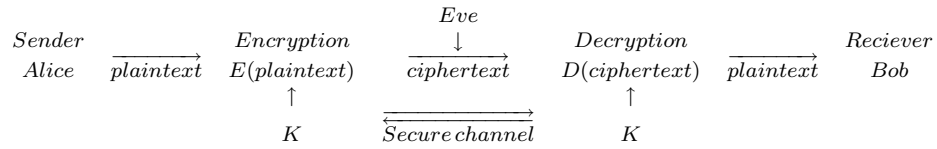

Figure 2.1: The secure communication between Alice and Bob, over an insecure channel, where Eve is listening in.

situation is described in Figure 2.1. This kind of cryptosystem is in the literature referred to as a symmetric, one-key, private-key, conventional or secret-key cryptosystem.

Another cryptosystem is the public-key cryptosystem, often called an asymmetric cipher or a two-key cipher. In two-key ciphers each user has a public key for encryption, so that anyone can encrypt, and one private key for decryption, so that only the owner of the keys can decrypt. This feature simplifies the key management, since there is no need to exchange a key over a secure channel. The reason why these cryptosystems are not used for all encryption is that the public key cryptosystems known today are slow compared to secret key systems. This thesis will only consider symmetric key cryptosystems.

2.1.1 Design Principles

Diffusion and confusion are mentioned by Shannon [41] as two design principles. These principles are generally accepted, and Massey [28] interprets these principles as follows.

Confusion "The ciphertext statistics should depend on the plaintext statistics in a manner too complicated to be exploited by the cryptanalyst"

Diffusion "Each digit of the plaintext and each digit of the secret key should influence many digits of the ciphertext"

In addition to these universal principles of how to design a cipher, Shannon also discusses two more specific design principles.

1. Make the security of the system reducible to some known difficult problem.
2. Make the cipher secure against all known attacks.

Many public-key ciphers have security based on a well known problem, e.g. RSA[39], where it is assumed that to break RSA it is necessary to factor a large number. This principle is not used in the design of secret-key ciphers, for these ciphers the second principle is the best known design principle today.

2.1.2 Block Ciphers

There are two kinds of secret key ciphers; stream ciphers and block ciphers. In stream ciphers a long sequence of key bits are generated and exclusive or'ed with the plaintext. In block ciphers the plaintext is divided into blocks of a fixed length and encrypted into blocks of ciphertext using the same key. This thesis will consider only block ciphers. The mathematical definition of a block cipher is:

Definition 2.1.2 *[33] An n-bit block cipher is a function $E : V_n \times K \rightarrow V_n$ such that for each key $k \in K$, $E(p,k)$ is an invertible mapping (encryption function for k) from V_n to V_n, written $E_k(P)$. The inverse mapping is the decryption function, denoted $D_k(C)$. $C = E_k(P)$ denotes that ciphertext C results from plaintext P under k. The variable V_n is the space containing all the possible bit strings of length n.*

An n-bit block cipher with a fixed key is a permutation $p : GF(2^n) \rightarrow GF(2^n)$. It would require $log_2(2^n!) \approx (n - 1.44)2^n$ bits to represent the key such that all permutations p were possible, or roughly 2^n times the number of bits in a cipher block. With an ordinary block size, e.g. 64 bits, this is a much too big number for practical use, therefore the key size in a practical block cipher is much smaller, typically 128 bits or 256 bits.

A good encryption function must contain some non-linear component, and this is often a substitution box or s-box. An s-box is defined as a mapping $GF(2^n) \rightarrow GF(2^m)$, usually defined by a $n \times m$ lookup table.

Almost all block ciphers used today are iterated block ciphers. These ciphers are based on iterating a function several times, each iteration is called a round. A more formal definition of iterated block ciphers is given.

Definition 2.1.3 *In an r-round iterated block cipher the ciphertext is computed by iteratively applying a round function g to the plaintext, s.t.*

$$C_{i+1} = g(C_i\,,\, K_i),\ i = 1, ..., r$$

where C_i is the ciphertext after i rounds, and K_i is the ith round key.

The round key is derived from the cipher key by a key schedule, which is an algorithm that expands the master key or the cipher key. The cipher key is usually between 40 and 256 bits for a block cipher, and for an r-round iterated cipher this is expanded into r-round keys.

The round function is usually a combination of substitution and transposition. Substitution is when a block in the plaintext is substituted with another block by some substitution rule. Transposition is to permute the blocks or characters in the plaintext. In earlier ciphers substitution and transposition were used on their own as a cipher, where each plaintext symbol was a block, but this proved to be insecure because of the small block size. Most modern ciphers are a combination of substitution and transposition, and are often called product ciphers.

The Data Encryption Standard (DES) [34] has been the most widely used iterated block cipher since it was published in 1977, but it is now replaced by the Advanced Encryption Standard (AES) because of too small key and block size. The DES can be seen as a special implementation of a Feistel cipher, named after Horst Feistel [13], where the input to each round is divided into two halves, as in the following definition.

Definition 2.1.4 *A Feistel cipher with block size $2n$ and with r rounds is defined as follows.*

The round function is defined:

$$\begin{aligned} g : GF(2)^n \times GF(2)^n \times GF(2)^m &\rightarrow GF(2)^n \times GF(2)^n \\ g(X, Y, Z) &= (Y,\, F(Y, Z) + X) \end{aligned}$$

where F can be any function taking two arguments of n bits and m bits respectively and producing n bits. '+' is a commutative group operation on the set of n-bit blocks. We will assume that '+' is the bitwise exclusive-or operation, if not explicitly stated otherwise.
Given a plaintext $P = (P^L||P^R)$ and r round keys K_1, K_2, ..., K_r the ciphertext $C = (C^L||C^R)$ is computed in r rounds. Set $C_0^L = P^L$ and $C_0^R = P^R$ and compute for $i = 1, 2, ..., r$

$$(C_i^L||C_i^R) = (C_{i-1}^R||F(C_{i-1}^R, K_i) + C_{i-1}^L$$

Set $C_i = (C_i^L||C_i^R)$ and $C^L = C_r^R$ and $C^R = C_r^L$. The round keys $(K_1, K_2, ..., K_r)$, where $K_i \in GF(2)^m$, are computed by a key schedule algorithm on input a master key K.

The application of the Feistel structure guarantees that encryption and decryption are similar processes, independent of the exact specification of the so-called F-function. With the adoption of the Feistel structure, the design of the F-function can concentrate completely on the desired propagation properties without restrictions imposed by the invertability. Because of the success of the DES, many ciphers proposed in the two last decades are Feistel ciphers. A special class of Feistel ciphers are the DES-like ciphers, named after DES.

Definition 2.1.5 *A DES-like cipher is a Feistel cipher where the F function is defined as follows:*

$$\begin{aligned} F(C_i, K_i) &= f(g(C_i) + K_i) \\ f : GF(2^m) &\rightarrow GF(2^n),\ m \geq n \\ g : GF(2^n) &\rightarrow GF(2^m),\ \textit{an affine expansion mapping} \end{aligned}$$

There has also been proposed a generalized Feistel network described in [36], more about this in Section 5.5. Many new ciphers have bijective components only, which in Definition 2.1.5 would mean that $m = n$. This gives good distribution of difference, which is preferable when constructing ciphers with provable security against differential and linear attacks (see Chapter 4). Also, when a cipher has only bijective components, we can easily construct the decryption algorithm, since every bijective function has an inverse. A function f is bijective if $f(a) = f(b) \Rightarrow a = b$.

2.2 Cryptanalysis

So why is it so important to do cryptanalysis? The goal in cryptology is to create secure systems, where only the legitimate users should be able to learn anything about the plaintext from the ciphertext. So why do we try to break the cipher? It is because we want to try to make sure that to break the cipher is not possible, or at least computational hard. By computational hard we mean that it will take too much time or other resources to break the cipher. Only when we have tried all thinkable attacks on the cipher, can we begin to trust the security of the cipher. Even a very theoretical attack on a cipher, an attack that perhaps is not possible to do in a practical way, but reduces the complexity of the cipher, usually results in that the cipher will not be recommended for further use.

Preferably we would like to be able to give a proof of the security of a cipher, but the problem with provable security for a block cipher is that if we assume that it is possible to recognize the plaintext, then a block cipher is always totally breakable in after observing some ciphertexts. Assume that the plaintext is not random, e.g. English text encoded with ASCII characters. The secret key can easily be found, simply by trying all possible keys one by one and check whether the computed text is meaningful. It is possible to get meaningful text for more than one key in this attack, and then we need more than one ciphertext block to uniquely distinguish the correct key. This attack requires computation of about 2^k encryptions, where k is the number of bit in the key. It is also possible to precompute a table of encryptions of a fixed plaintext P under all possible keys and sort and store the ciphertext. Thereafter the cipher is totally breakable if we can obtain the ciphertext of a chosen plaintext. These attacks goes under the name brute-force or exhaustive search attack.

This motivates the notion of computational security. The way to prevent these trivial attacks is to choose a key big enough so that it is computational impossible to do brute force attacks. This leads to a more practical definition of a broken cipher, we say that a cipher is broken if it is possible to find (some bits of the) key with less work than an exhaustive search for the key. So how big must a key be to prevent brute force attacks? We have the following rule of thumb:

Definition 2.2.1 *(Moores Law) The computing power for a given cost is multiplied by four every 3 years.*

The DES with a 56 bits key was broken in 22 hours in a distributed key search on the Internet, and on Crypto2001 Jean-Jacques Quisquater broke 6 out of 8 DES-keys with 40 unknown bits on a small laptop during a 3 minutes rump-session talk, using Hellman's memory trade-off[15]. In the late 1990s, specialized "DES Cracker" machines were built that could recover a 56-bits DES-key after a few hours. In other words, by trying all possible key values, the hardware could determine which key was used to encrypt a message. The AES has block length 128 bits and three different key lengths 128, 196 and 256 bits, this is assumed to be sufficient for the foreseeable future[27].

Breaking a cipher is not necessarily to find a practical way for an eavesdropper to recover the plaintext from the ciphertext or to find the key. In academic cryptology a cipher is said to be broken, if there has been found a weakness in the cipher that can be exploited with less complexity than a checking all keys. We would like to make the security of a cipher dependent only on the key, so we make the following assumption.

Kerckhoffs' Assumption ([18]) The enemy cryptanalysis knows all details of the enciphering process and deciphering process except for the value of the secret key.

This is in many cases a proper assumption to make, software can be reverse engineered and tamper proof hardware can be opened and details compromised. It is also possible that one of the designers of the cipher can be the attacker. The most important aspect with this design principle is that it removes the trust from the designers and place it solely on the cipher.

Other assumptions we can make are that the attacker has access to an encryption oracle, which encrypts any given plaintexts, or perhaps only encrypts some plaintexts, or decrypts ciphertexts. This motivates the following classification of attacks.

2.2.1 Classification of Attacks

We can classify the possible attacks an attacker can do [42].

Ciphertext only attack The attacker possesses a set of intercepted ciphertexts.

Known plaintext attack The attacker obtains a set of s plaintexts P_1, P_2, ..., P_s and the corresponding ciphertexts C_1, C_2, ..., C_s. That is, the attacker has no control over the pairs of plain- and ciphertexts available to him.

Chosen plaintext attack The attacker chooses a priori a set of s plaintexts P_1, P_2, ..., P_s and obtains in some way the corresponding ciphertexts C_1, C_2, ..., C_s. That is, he has a encryption oracle.

Adaptively chosen plaintext attack The attacker chooses a set of s plaintexts P_1, P_2, ..., P_s interactively as he obtains the corresponding ciphertexts C_1, C_2, ..., C_s. That is, the attacker chooses P_1, obtains C_1, then chooses P_2 etc.

Chosen ciphertext attacks For symmetric ciphers these are similar to those of chosen plaintext attack and adaptively chosen plaintext attack, where the roles of plain- and ciphertexts are interchanged.

The weakest attack here is ciphertext only, which is also the most likely attack. But there are situations where the other attacks are just as likely, e.g. encryption with a smart card. Ideally the designer of a cipher should prove that the system is secure against an adaptively chosen plaintext/ciphertext attack, then it would be secure against the other types of attacks as well.

2.2.2 Classification of Breaking Ciphers

There are other ways to break a cipher than just finding the key.

Total Break An attacker finds the secret key K.

Global Deduction An attacker finds a formula functionally equivalent to $E_K(\cdot)$ (or $D_K(\cdot)$) without knowing the key K.

Instance (local) deduction An attacker finds the plaintext (ciphertext) of an intercepted ciphertext (plaintext), which he did not obtain from the legitimate sender.

Information deduction An attacker gains some information about key or plaintexts from the ciphertexts, which he did not get directly from the sender and which he did not have before the attack.

In the total break, also called key-recovery attack, the goal is to find the key. The attacks in this thesis are distinguisher attacks. The distinguisher is some information about the ciphertext after r rounds of a cipher, and it can be used to attack $r+1$-round of the cipher. We guess the last round key, or some bit of it, and decrypt the ciphertext one round. Now we use our distinguisher to see if our guess is correct. This may have to be repeated to uniquely distinguish the right key. When the last round key is found, similar attacks to this can be mounted on a cipher one round shorter.

2.2.3 Complexity of Attacks

It is natural to judge an attack on a cipher after how much effort goes into a breaking it. Important factors in an attack are how much data we have to gather before we can break the system, how long time an attack takes and how large the storage requirements are.

Data complexity How many plaintext/ciphertext are needed as input in the attack.

Processing complexity How long time does the attack take. Time in this setting is measured in how many encryption/decryption that are needed.

Storage requirements How much memory is needed in the attack. Units are measured in blocks of length n.

Trade-offs are often possible here. For example time-memory trade-off where doing some calculations in advance and storing the results can lower the processing requirements during the actual attack. One example of this is Hellman's time-memory trade-off attack[15], which is applicable to any block cipher. This attack finds the secret k-bit key after $2^{2k/3}$ encryptions using $2^{2k/3}$ words of memory. The $2^{2k/3}$ words of memory are pre-computed, and takes time equivalent to 2^k encryptions.

Chapter 3

Some Attacks on Block Ciphers

In this section we are going to look briefly at some of the well known cryptanalytic attacks on block ciphers.

3.1 Differential Cryptanalysis

Differential cryptanalysis was first published by Eli Biham and Adi Shamir [6], even though it is assumed that it was known as early as the 1970s when the DES was developed. The resilience of the s-boxes in the DES to differential attack indicates that this attack was known by the designers of the DES, but the design criteria have been classified by the National Security Agency (NSA). This was the first attack which could recover DES-keys with less complexity than exhaustive search, and several other ciphers have also been broken by differential cryptanalysis, e.g. FEAL [5]. This attack has led to the redesign of many ciphers.

This attack is a statistical, chosen plaintext attack, that exploits the correlation between input and output differences. The idea of differential cryptanalysis is to choose a pair of plaintexts with a known difference and try to predict the difference of the pair after some rounds of an iterated cipher. When we have this prediction for $r-1$ rounds we can do an attack on r rounds by guessing some key bits in the last round, calculate backwards and see if we get the predicted difference after $r-1$ rounds. The main criterion for a differential attack to be successful on an iterated cipher with some non-linear component F, is that the distribution of differences through F is non-uniform, which means that some output differences have to be more likely than others.

3.1.1 Some Definitions

Assume that the key K is combined to the text, X, via some group operation $\otimes$, then we can define difference:

Definition 3.1.1 *The difference between X_1 and X_2 is defined as:*

$$\Delta(X_1, X_2) = X_1 \otimes X_2^{-1}$$

The advantage of this definition of difference is easily seen when we look at how the key combination effects the difference:

$$\begin{aligned}\Delta(X_1 \otimes K,\, X_2 \otimes K) &= (X_1 \otimes K) \otimes (X_2 \otimes K)^{-1} \\ &= X_1 \otimes K \otimes K^{-1} \otimes X_2^{-1} \\ &= \Delta(X_1,\, X_2)\end{aligned}$$

Definition of difference is relative to the cipher, but in most ciphers the difference is the exclusive-or or xor (notation: $\oplus$) of the texts. This makes it very easy to find inverses, since $x_2 \oplus x_2 = 0$ and then $x_2^{-1} = x_2$. We denote the difference between two plaintexts and the corresponding ciphertexts by ΔP and ΔC respectively.

Definition 3.1.2 *[6] An s-round characteristic is a series of differences defined as an $(s+1)$-tuple*

$$\Omega : \{\alpha_0,\, \alpha_1,\, ...,\, \alpha_s)$$

where $\Delta P = \alpha_0$, $\Delta C_i = \alpha_i$ for $1 \leq i \leq s$

We judge how "good" a characteristic is by how high probability it has

$$Pr_{K,P}(\Omega) = Pr_{K,P}(\Delta C_s = \alpha_s,\, ...,\, \Delta C_1 = \alpha_1 | \Delta P = \alpha_0) \tag{3.1}$$

but this probability can be difficult to compute because it is over all possible plaintexts and keys. The calculation of a probability of a characteristic is easier for a Markov cipher.

Definition 3.1.3 *[26] An iterated cipher is called a Markov cipher, if there is a group operation $\otimes$ (defining Δ), such that*

$$Pr(\Delta C_1 = \beta | \Delta C_0 = \alpha,\, C_0 = \gamma)$$

is independent of γ for all α, β, when the round key K is chosen uniformly at random.

To calculate the probability of a characteristic for a Markov cipher with independent round keys it is enough to find the probability for each round and multiply them together:

$$\begin{gathered}Pr_{K,P}(\Delta C_s = \alpha_s,\, ...,\, \Delta C_1 = \alpha_1 | \Delta P = \alpha_0) = \\ Pr_K(\Delta C_s = \alpha_s,\, ...,\, \Delta C_1 = \alpha_1 | \Delta P = \alpha_0) = \\ \textstyle\prod_{i=1}^{s} Pr_K(\Delta C_1 = \alpha_i | \Delta P = \alpha_{i-1})\end{gathered}$$

Fact: The DES is a Markov cipher with difference exclusive-or.

3.1.2 Differentials

When we take a closer look at the differential attack, we see that for an s-round characteristic $(\Delta P, \Delta C_1, ..., \Delta C_s)$ only the plaintext difference ΔP and the last ciphertext difference ΔC_s make a difference in the attack. That is, for the attack itself it does not matter what the values of $\Delta C_1, ..., \Delta C_{s-1}$ are, they are not

used in the attack. The notion of differentials $(\Delta P, \Delta C_s)$ was first used by Lai and Massey [26, 24] to describe this observation. The probability of an s-round differential $(\Delta P, \Delta C_s)$ is the conditional probability that the output difference after s rounds is ΔC_s given ΔP.

Definition 3.1.4 *[19] The probability of an s-round differential is given as*

$$Pr(\Delta C_s = \beta_s | \Delta P = \beta_0) = \sum_{\beta_1} \sum_{\beta_2} \cdots \sum_{\beta_{s-1}} \prod_{i=1}^{s} Pr(\Delta C_i = \beta_i | \Delta C_{i-1} = \beta_{i-1})$$

when $\Delta C_0 = \Delta P$.

When we want to make an attack on a DES-like cipher, it is sufficient to find one differential with probability high enough to do a successful attack, but to make sure that a cipher is resistant against a differential attack, it is necessary to make sure that no differential with probability high enough to make it possible to mount a differential attack exists. Definition 3.1.4 shows that this is difficult, or computational hard, to calculate the probability of every differential. However, it is possible to give an upper bound on the probability of differentials for a given cipher. More on this is given in Chapter 4.

3.1.3 Principle of a General Attack

If we are going to mount a differential attack on an r-round iterated block cipher we need an $(r-1)$-round differential determining ΔC_{r-1} with probability p. We will then find the last round key K_r (or k bits of it) by using the following procedure:

1. Choose pairs of plaintexts P and P^* with difference ΔP.
2. Get the pairs of ciphertexts $C = E_K(P)$ and $C^* = E_K(P^*)$.
3. For $i = 0$ to $2^k - 1$ do
   - From ciphertexts and the guessed $K_r = i$, decrypt ciphertexts one round.
   - If the decrypted ciphertexts give the expected difference ΔC_{r-1}, the counter for i is incremented by 1.
4. Repeat steps $1 - 3$ until one counter has a value bigger than the other counters.

A right pair is a pair of plaintexts that follows each step of the characteristic.

A wrong pair is not a right pair.

The strategy for improving and in many cases making an attack possible, is to minimize the number of wrong pairs. It is often possible from the ciphertexts alone to determine that a pair is wrong, in that case the pair is filtered out and not used in the analysis.

3.1.4 Complexity

First we have to describe an important term in the analysis of complexity of differential attacks. The key we are looking for has to be suggested more (or less) times than any other wrong value of the key. Signal to noise ratio gives a description of how the key we are looking for can be recognized.

Definition 3.1.5 *Signal to noise ratio:*

$$S/N = \frac{\#\textit{times correct key is counted}}{\#\textit{times a random key is counted}}$$

k *number of key bits to find.*

p *probability of characteristic.*

m *number of pairs required.*

β *ratio of used pairs to all pairs.*

α *number of keys suggested by each used pair of ciphertexts.*

$$S/N = \frac{mp}{\frac{m\beta\alpha}{2^k}} = \frac{p2^k}{\beta\alpha}$$

If $S/N = 1$ a differential attack is impossible, and if $S/N > 1$ the most suggested key is the right key and if $S/N < 1$ the least suggested key is what we look for.

In a differential attack the number of chosen plaintexts needed is about:

$$\frac{c}{p_\Omega}$$

where p_Ω is the probability of the differential used and $c > 1$ is a function of S/N. The success of the attack depends on probability of characteristic, S/N ratio (which can be increased by filter out wrong pairs), numbers of counters required and time to run the attack, i.e. how many plaintext pairs we need and how many key bits we are looking for. The attack on the full 16-round DES needs about 2^{47} chosen plaintexts to succeed.

3.2 Linear Cryptanalysis

Linear cryptanalysis was first introduced by Matsui in 1993 [31], and it is a known plaintext attack and the best known attack on the DES today. The main idea here is to use linear relations between bits in plaintext P and ciphertext $C = E_K(P)$ to determine (bits of) key K.

3.2.1 Introduction

If we let C_i be the ciphertext after i rounds, we can write a one-round linear approximation as

$$(C_i \cdot \alpha) \oplus (C_{i+1} \cdot \beta) = 0 \tag{3.2}$$

which holds with a certain probability p, where C_i, C_{i+1}, α, β are m-bits strings and '$\cdot$' is the dot (or inner) product modulo 2. α, β are often called masks. $|p_i - 1/2|$ is called the bias of the approximation

Note The Expression 3.2 with a '1' on the right side will have a probability of $1-p$, but the bias for the two expressions are the same.

The following describes why bias is a better way to describe the linear relation between input and output of a round of a cipher when we do not know the value of the key k. Consider the DES-like cipher

$$\begin{array}{cccc} & & k_i & \\ & & \downarrow & \\ C_i \rightarrow & \oplus & \rightarrow D \rightarrow f \rightarrow & C_{i+1} \end{array}$$

We have that C_i is the ciphertext after i rounds, and k_i is the ith round key. D is C_i been exclusive-or'ed with the key k_i. Then D is inputed to the non-linear function f which together with the exclusive-or with the round key is the round function. The output C_{i+1} is the ciphertext after $i+1$ rounds. Now we want to make the linear approximation of this round as

$$\begin{array}{rcl} (\alpha \cdot C_i) & \oplus & (\alpha \cdot D) = (\alpha \cdot k_i) \\ (\alpha \cdot D) & = & (\beta \cdot C_{i+1}) \text{ with } p_i \neq 1/2 \\ (\alpha \cdot C_i) & \oplus & (\alpha \cdot k_i) = (\beta \cdot C_{i+1}) \text{ with } p_i \neq 1/2 \\ (\alpha \cdot C_i) & = & (\beta \cdot C_{i+1}) \text{ with bias } |p_i - 1/2| \end{array}$$

These 1-round linear relations are then put together into an r-round characteristic $(\delta_0, ..., \delta_r)$. To calculate the probability of linear characteristics the piling-up lemma [31] is used.

Lemma 3.2.1 *(Piling-Up Lemma [31]) Let $Z_i, 1 \leq i \leq n$, be independent random variables, whose boolean values are 0 with probability p_i. Then*

$$Pr(Z_1 \oplus \ldots \oplus Z_n = 0) = 1/2 + 2^{n-1} \prod_{i=1}^{n} (p_i - 1/2)$$

This gives us the probability $p_L = 1/2 + 2^{r-1} \prod_{i=1}^{r} (p_i - 1/2)$ for the r-round linear characteristic, which again gives us the bias $|p_L - 1/2| = \left|2^{r-1} \prod_{i=1}^{r} (p_i - 1/2)\right|$.

3.2.2 Principle of a General Attack

The attack on an r-round DES-like iterated cipher can be described by the following procedure:

1. Find an $(r-1)$-round characteristic $(\delta_0, ..., \delta_{r-1})$:

   $$(P \cdot \delta_0) \oplus (C_{r-1} \cdot \delta_{r-1}) = 0$$

   with some bias $b \neq 0$.

2. Obtain the plaintexts with corresponding ciphertexts.

3. Then make a guess for a value of the last round key, K_r, and calculate one round back from the ciphertexts:

   $$(P \cdot \delta_0) \oplus (f^{-1}(C, K_r) \cdot \delta_{r-1}) = 0 \tag{3.3}$$

   with bias b'.

(a) For right value of K_r, (3.3) is the linear characteristic for $(r-1)$ rounds, and in this case the bias $b' = b$.

(b) Assumption: for wrong values of K_r, (3.3) is a random approximation with bias $b' \simeq 0$.

4. Repeat step 3 for all values of K_r, and increase a counter i for each time K_i gives the right value in (3.3).

5. The value for K_r that produces bias closest to expected is the key we are looking for.

It is also possible to try to find key bits in both the first and last round of the cipher.

3.2.3 Complexity of the Attack

The complexity, N_p, of the attack on the DES with up to 16 rounds is estimated to be

$$N_p \simeq c \times |p_L - 1/2|^{-2}$$

where $c \leq 8$. The attack on the DES requires about 2^{43} plaintexts.

3.3 Higher Order Differentials

Some ciphers with proof of resistance against differential and linear attacks can successfully be cryptanalysed using higher order differentials.

Lai gives the definition of derivatives of discrete functions in [25].

Definition 3.3.1 *[25] Let $(S,+)$ and $(T,+)$ be Abelian groups. For a function $f : S \to T$, the derivative of f at the point $a \in S$ is defined as*

$$\Delta_a f(x) = f(x+a) - f(x)$$

Definition 3.3.2 *[25] Let f be as in Definition 3.3.1. The i'th derivative of f at the point $a_1, a_2, ..., a_i$ is defined as*

$$\Delta^{(i)}_{a_1,...,a_i} f(x) = \Delta_{a_i}(\Delta^{(i-1)}_{a_1,...,a_{i-1}} f(x))$$

First order derivatives using definition by Lai coincides with the original differentials and characteristics as defined by Biham and Shamir and used in their attacks. We can extend the notion of differentials into higher order differentials.

Definition 3.3.3 *[19] An 1-round differential of order i is an i+1-tuple $(\alpha_1, ..., \alpha_i, \beta)$, s.t.*

$$\Delta^{(i)}_{a_1,...,a_i} f(x) = \beta$$

That is, we see how the difference between i inputs to round function f, the ciphertexts C_{r-1}, go through round r, and the difference between the i outputs, the ciphertexts C_r, are then some β. To be able to say more about how many ciphertexts that are needed in each differential, or in other words, what is the best order of a higher order differential, we need some more results.

Proposition 3.3.1 *[25] Let $L[a_1, a_2, ..., a_i]$ be the list of all 2^i possible linear combinations of $a_1, a_2, ..., a_i$. Then*

$$\Delta^{(i)}_{a_1,...,a_i} f(x) = \sum_{\gamma \in L[a_1,...,a_i]} f(x \oplus \gamma)$$

If a_i is linearly dependent of $a_1, ..., a_{i-1}$ then

$$\Delta^{(i)}_{a_1,...,a_i} f(x) = 0$$

Another important proposition from [25] is

Proposition 3.3.2 *[25] Let $ord(f)$ denote the non-linear order of a multivariable polynomial function $f(x)$. Then*

$$ord(\Delta_a f(x)) \leq ord(f(x)) - 1$$

This gives us the following proposition.

Proposition 3.3.3 *[19] If $\Delta_{a_1,...,a_i} f(x)$ is not a constant, then the non-linear order of f is greater than i.*

Knudsen[19] shows that we generally can attack a cipher with five rounds, where the round function has nonlinear order r, using r'th order differentials. This indicates that ciphers should not use round function of low non-linear order.

3.4 Truncated Differentials

An other variation of the differential attack is truncated differentials, and it is due to an idea by Knudsen [21].

Definition 3.4.1 *[21] A differential that predicts only parts of an n-bit value is called a truncated differential. More formally, let (a, b) be an i-round differential. If a' is a subsequence of a and b' is a subsequence of b, then (a', b') is called an i-round truncated differential.*

A truncated differential is a collection of ordinary differentials, only using a part of the difference between the texts of the ordinary differential. This allows the rest of the bits in the difference to chosen freely, which gives a higher probability for truncated differentials compared to differentials.

Knudsen [22] shows an attack on a 5-round version of SAFER K-64 [29] with truncated differentials. SAFER K-64 is resistant against a conventional differential attack after 5 rounds [30] and against linear attack after 2 rounds [14].

3.5 Interpolation Attack

This is an attack introduced by Jakobsen and Knudsen [17] on ciphers using simple algebraic functions as s-boxes. This attack is based on a Lagrange interpolation formulas (see e.g. [8]).

Let R be a field. Given $2n$ elements $x_1, ..., x_n, y_1, ..., y_n \in R$, where $x_i \neq x_j,\ i \neq j$. Define

$$f(x) = \sum_{i=1}^{n} y_i \prod_{1 \leq j \leq n, j \neq i} \frac{x - x_j}{x_i - x_j}$$

Then $f(x)$ is the only polynomial over R of degree at most $n-1$ such that $f(x_i) = y_i \forall i = 1, ..., n$.

3.5.1 Global and Instance Deduction

Jakobsen and Knudsen constructed a cipher called $\mathcal{PURE}$, secure against differential [38] and linear [37] attacks. The round function in the DES-like Feistel cipher is $F_k(x) = f(x \oplus k)$ where $f : GF\left(2^{32}\right) \rightarrow GF\left(2^{32}\right),\ f(x) = x^3$.

In [17] a global deduction attack is mounted on an r-round version of $\mathcal{PURE}$. Keeping in mind that addition over a finite field with characteristic 2 is the same as exclusive or ($\oplus$), we have that the cipher only consists of simple algebraic operations. This means that each of the halves of the ciphertext y can be described as a polynomial $p\left(x_L, x_R\right) \in GF\left(2^{32}\right)\left[x_L, x_R\right]$ of the plaintext with at most $3^{2r-1} + 3^r + 3^{r-1} + 1$ coefficients, since the degree of x_L and x_R are at most 3^{r-1} and 3^r respectively. Using the Lagrange Interpolation Formula, we can reconstruct this polynomial considering at most $3^{2r-1} + 3^r + 3^{r-1} + 1$ plaintext/ciphertext pairs. With $r = 6$ the attack needs at most 2^{18} known plaintext/ciphertext pairs, which gives us an algorithm for a global deduction. The actual number of coefficients will be lower than specified, since not all elements $x_L^i x_R^j$ for $0 \leq i \leq 3^{r-1}$ and $0 \leq j \leq 3^r$ will appear in the polynomial. More generally:

Theorem 3.5.1 *[17] Consider an iterated block cipher with block size m. Express the ciphertext as a polynomial of the plaintext and let n denote the number of coefficients in the polynomial. If $n \leq 2^m$, then there exists an interpolation attack of time complexity n requiring n known plaintexts encrypted with a secret key K, which finds an algorithm equivalent to encryption (or decryption) with K.*

3.5.2 Key-recovery

This method can be extended to a key-recovery attack, since we have a distinguisher that can tell the difference between the output from the cipher after r rounds from a random m bit string. Now we express the output from the reduced cipher as a polynomial $p(x) \in GF\left(2^m\right)[x]$ of the plaintext. Assuming that the polynomial has degree d and that $d+1$ known plaintext/ciphertext pairs are available. Then the attack goes:

1. For all values of the last round key
   (a) Decrypt the ciphertexts one round and try to construct the polynomial.
   (b) Use one extra plaintext/ciphertext pair to check whether the polynomial is correct.

2. When the polynomial is correct, the last round key has been found with high probability.

We have the following theorem:

Theorem 3.5.2 *[17] Consider an iterated block cipher of size m. Express the output from the round next to the last as a polynomial of the plaintext and let n denote the number of coefficients in the polynomial. Furthermore. let b denote the number of last round key bits. Then there exists an interpolation attack of average time complexity $2^{b-1}(n+1)$ requiring $n+1$ known (or chosen) plaintexts which will successfully recover the last round key.*

3.6 Other Attacks

We have looked at differential attacks, linear attacks, higher order and truncated differentials. Other types of attacks are exhaustive search, related keys, Slide attack [7] by Biryukov and Wagner and Boomerang Attack [43] by Wagner, impossible differentials [3][4] and integral attacks. Integral attacks is a "family" of attacks that exploits the fact that modern n-bit block ciphers often use permutations: $p : \{0,1\}^w \rightarrow \{0,1\}^w$ with $w < n$ as building blocks. E.g., p can be an s-box, a round function or a group operation where one of the operands is constant (e.g. a secret key). An integral attack is based on the idea of putting every possible input to such a permutation p equally many times. We can then say that p is saturated, and since p is a permutation, or a bijective function, we have that the output is saturated as well. That is the reason why some have called this attack for saturation attack. Chapter 5 provides more details about integral attacks.

Chapter 4

Resistance Against Differential Attacks

Lately several ciphers have appeared with proof of resistance against differential attack. This idea was first introduced by Nyberg and Knudsen in [38], and they gave an example of a cipher with provable security against differential attack. Later Matsui has proposed MISTY [32] and KASUMI[1] with this property.

4.1 Some Results

We will look at an r-round iterated block cipher with round function f, and we will use p_f to denote the highest probability of a non-trivial 1-round differential:

$$p_f = \max_{\beta} \max_{\alpha \neq 0} Pr_K(\Delta C_1 = \beta | \Delta P = \alpha) \tag{4.1}$$

where the probabilities are taken over all possible keys. We can now obtain a lower bound of any differential in an r-round iterated cipher expressed in terms of p_f.

Theorem 4.1.1 *[19] Consider an r-round iterated cipher with independent round keys. Any s-round differential, $s \geq 1$, has a probability of at most p_f, where p_f is the probability of the most likely 1-round differential*

Theorem 4.1.1 is trivial when it comes to the DES-like ciphers, since if the right halves of a pair is equal, then $p_f = 1$. These differentials are called trivial 1-round differentials for DES-like ciphers. It is possible to calculate a lower bound on all differentials for a DES-like iterated cipher expressed in terms of the most likely non-trivial 1-round differential. Let p_{max} denote:

$$p_{max} = \max_{\beta} \max_{\alpha_R \neq 0} Pr(\Delta C_1 = \beta | \Delta P = \alpha) \tag{4.2}$$

where α_R is the right half of α. From now on we assume that $p_{max} < 1$.

Theorem 4.1.2 *[19] Consider an r-round iterated DES-like cipher with independent round keys. Any s-round differential, $s \geq 4$, has a probability of at most $2p_{max}^2$.*

If the round function F of a DES-like cipher is a permutation for each given round key, we say that the F-function is a permutation. In this case Theorem 4.1.2 can be proven for $s \geq 3$.

Theorem 4.1.3 *[19] Consider an r-round iterated DES-like cipher with independent round keys, where the F-function is a permutation. Any s-round differential, $s \geq 3$, has a probability of at most $2p_{max}^2$.*

This has later been improved to p^2 by Aoki and Ohta [2].

There are similar results for linear characteristics.

4.2 Conclusion Of This Chapter

The results in the previous section make it possible to design ciphers where it is possible to obtain an upper bound on the probability, p, of any differential characteristic for the cipher. It is then likely to conclude that it will take at least $1/p$ chosen plaintexts/ciphertexts pairs to break the cipher with a differential attack. This is what most designers of block ciphers do, but sadly this is not sufficient to say that a cipher is immune to differential attacks. Biham et al.'s impossible differentials [3][4] use differentials of sufficiently low probability in attacks on ciphers. Wagner's Boomerang Attack [43] also shows that the provable resistance against differential attack is not enough to say that it will require at least $1/p$ texts to break the cipher with differentials.

To focus too much on making a cipher secure against one attack is dangerous, because it might make the cipher vulnerable to other attacks. An example of this is the Feistel cipher $\mathcal{PURE}$ [17] where the round function $F_k(x) = f(x \oplus k)$ where $f : GF(2^{32}) \rightarrow GF(2^{32})$, $f(x) = x^3$. This cipher is secure against differential attack [38] and linear attack [37], but breakable by interpolation attack (see Section 3.5).

Chapter 5

Integral Cryptanalysis of Block Ciphers

Integral cryptanalysis is a chosen plaintext attack, and it has much in common with differential attacks. However, it applies to ciphers not vulnerable to differential attacks. It is especially applicable to ciphers using bijective components only. It is a chosen plaintext attack and was first introduced by Daemen, Knudsen and Rijmen as a special attack on the Square cipher[9]. It has been known as the Square attack or the saturation attack, but on Fast Software Encryption 2002 Knudsen and Wagner[23] presented a generalization and introduced the term *integral cryptanalysis* about this attack.

5.1 Introduction

Definition 5.1.1 *[23] Let $(G, +)$ be a finite Abelian group of order k. Consider the product $G^n = G \times ... \times G$, that is, the group with elements of the form $v = (v_1, ..., v_n)$ where $v_i \in G$. The addition of G^n is defined component-wise, so that $u + v = w$ holds for $u, v, w \in G$ just when $u_i + v_i = w_i$ for all i*

Let S be a multi set of vectors. An integral over S is defined as the sum of all vectors in S. In other words, the integral is

$$\int S = \sum_{v \in S} v$$

where the summation is defined in terms of the group operation for G. (For a multiplicative group this would usually be called a "product").

In integral cryptanalysis we divide the plaintexts and ciphertext into smaller blocks, which we call words. A word in this context is just a bit string of a given length s. If n represents the number of words in the plaintext and ciphertexts, and m denotes the number of plaintexts and ciphertexts considered (at a time), then typically, $G = GF(2^s)$ or $G = Z/kZ$, $m = k = 2^s$ (recall that $k = |G|$), and the vectors $v \in S$ represent the plaintext and ciphertexts.

When we want to do an integral attack, we must try to predict something about the value of the integral after a certain number of rounds of encryption.

We distinguish between three different cases; where all ith words are equal, all different, or sum to a certain value. Let S be as before, then we consider these cases:

$$v_i = c \quad \text{for all } v \in S \tag{5.1}$$

$$\{v_i : v \in S\} = G \tag{5.2}$$

$$\sum_{v \in S} v_i = c' \tag{5.3}$$

where $c, c' \in G$ are some known values that are fixed in advance.

If we now consider the case where $m = k$, that is the number of vectors in the set S is equal to the number of elements in the considered group. If all the ith words are equal then it is obvious that the ith word in the integral will take the value of the neutral element in G. In group theory there are a result that makes it easy to calculate the integral in the second case [16, Problem 2.1, p. 116]:

Theorem 5.1.1 *[16] Let G be a finite Abelian additive group, and let $H = \{g \in G : g + g = 0\}$ be the subgroup of elements of order 1 or 2. Write $s(G)$ for the sum $\sum_{g \in G} g$ of all the elements of G. Then $s(G) = \sum_{h \in H} h$. Moreover $s(G) \in H$, i.e. $s(G) + s(G) = 0$.*

Proof: Assume that G is a finite Abelian additive group, and $H = \{g \in G : g + g = 0\}$ is a subgroup of G. Then $s(G) = \sum_{g \in G} g = g_1 + ... + g_n$. For those $g_i, 1 \leq i \leq n$ that have $order(g_i) > 2$ we have that $\exists g_j, 1 \leq j \leq n, j \neq i$ such that $g_i + g_j = 0$. Then, there only remains the elements that have order 1 or 2 in the sum $s(G)$, and hence $s(G) = \sum_{h \in H} h$. $s(G) \in H$ since H is closed under the group operation of G, and $s(G)$ is a sum of elements from H.

If we look at $G = GF(2^s)$ we get $s(G) = 0$. To see this, observe that all elements in $GF(2^s)$ have order 2, and that $+$ is equal to x-or in this group. If $G = Z/mZ$ we get $s(G) = m/2$ if m is even and $s(G) = 0$ if m is odd. To see this, observe that only 0 and $m/2$ can have order 2 in this group. But if m is odd, then $m/2$ will not exist in Z/mZ. There is an analogue for multiplicative groups.

Theorem 5.1.2 *[16] Let G be a finite multiplicative group, and let $H = \{g \in G : g * g = 1\}$ be the subgroup of elements of order 1 or 2. Write $p(G)$ for the product $\prod_{g \in G} g$ of all the elements of G. Then $p(G) = \prod_{h \in H} h$. Moreover $p(G) \in H$, i.e. $p(G) * p(G) = 1$.*

Proof: Assume that G is a finite multiplicative group, and $H = \{g \in G : g * g = 1\}$ is a subgroup of G. Then $p(G) = \prod_{g \in G} g = g_1 * g_2 * ... * g_n$. For those $g_i, 1 \leq i \leq n$ that have $order(g_i) > 2$ we have that $\exists g_j, 1 \leq j \leq n, j \neq i$ such that $g_i * g_j = 1$. Then, there only remains the elements that have order 1 or 2 in the product $p(G)$, and hence $p(G) = \prod_{h \in H} h$. $p(G) \in H$ since H is closed under the group operation of G, and $p(G)$ is a product of elements from H.

For $G = (Z/mZ)^*$, the product of all the group elements depends on m, if m is prime then $p(G) = -1$ (Wilson's Theorem).

These two theorems give us tools to decide the value of the integral after it has gone through a component of the cipher, in all of the above three cases ((5.1), (5.2) and (5.3)).

In differential cryptanalysis over a group G, it is typically to define difference with subtraction or division, e.g. $dx = x' - x$ for an additive group or $dx = x' * x^{-1}$ for a multiplicative group. In [23] Knudsen and Wagner claim that the right operation for integrals is addition or multiplication. They base this claim on how integrals go through different parts of a block cipher.

If a cipher computes $w_j = u_j + v_j$ where u_j, v_j, w_j are intermediate values, and the integral predicts that the words u_j and v_j are on one of the forms (5.1), (5.2) or (5.3). Then we have

$$\sum_j w_j = \sum_j u_j + \sum_j v_j$$

and if we know the sum of the words u_j and v_j, we can determine the sum of the words w_j. So, if all the words u_j are equal and all the words v_j are different or visa-versa, all the words w_j will be different. And if all the words u_j are equal and v_j are equal, then so will all the words w_j.

All good ciphers contain non-linear components or non-linear s-boxes. Suppose that a function f is applied to a word in the cipher, i.e. $v_j = f(u_j)$. It is obvious that if all the words u_j are equal, then so will all the words v_j. We have that if f is a bijection or a permutation, and if all the words u_j are different then all the words v_j will also be different.

5.2 Higher Order Integrals and Some Notations

Just as we can define higher order differentials (see Section 3.3), we can define higher order integrals.

We look at a set $\widetilde{S} = S_1 \cup ... \cup S_s$ made up of s sets of vectors, where each S_i forms an integral. Then, if one can determine the sum of elements of S_i for each i, then one can also determine the sum of all vectors in $\widetilde{S}$. Suppose the words in a cipher can take m values. Consider a set of m vectors (representing a set of plaintexts) which differ only in one particular word. The sum over the vectors of this set is called a first order integral.

Definition 5.2.1 *[23] Consider a set of m^d vectors which differ in d components, such that each of the m^d possible values for the d-tuple of values from these components occurs exactly once. The sum of this set is called an dth order integral.*

We will use the same notation for words in an integral as in [23]. For the first order integral

'$\mathcal{C}$' (for "Constant") in the ith entry, means that all the values of all ith words in the collection of texts are equal.

'$\mathcal{A}$' (for "All") means that all words in the collection of texts are different.

'$\mathcal{S}$' (for "Sum") means that the sum of all ith words can be predicted.

'?' will be written when the sum of words can not be predicted.

For dth order integrals we use '$\mathcal{C}$' and '?' as before, and

$\mathcal{A}^{\mathrm{d}}$ means that the corresponding component participates in a dth order integral. That is, if we assume that one word can take m different values, then $\mathcal{A}^{\mathrm{d}}$ means that in the integral the particular word takes all values exactly m^{d-1} times. We use $\mathcal{A}$ as a short notation for $\mathcal{A}^1$.

$\mathcal{A}_{\mathrm{i}}^{\mathrm{d}}$ means that in the integral the string concatenation of all words with subscript i take the m^d values exactly once.

5.3 Comparison to Other Attacks

Integral cryptanalysis has similarities with both truncated differentials [20][21][22] and higher order differentials [21].

In truncated differentials (see Section 3.4) one is only often interested in if the words in a pair are equal or different. Integrals restricted to pairs of texts with only the values $\mathcal{C}$ and $\mathcal{A}$ are similar to such truncated differentials.

We also note that integrals are somewhat similar to higher order differentials. Recall the definition of first order derivative from Definition 3.3.1

$$f_a(x) = f(x+a) - f(x)$$

This is the definition used in a differential or characteristic which is traditionally used in cryptanalysis. This definition can be extended to a definition of higher order differentials. Also recall the definition of higher order derivatives Definition 3.3.2: The the ith-order derivative of f at the point $a_1, ..., a_i$ is:

$$f_{a_1,...,a_i}(x) = f_{a_i}(f_{a_{i-1},...,a_1}(x))$$

An example of this, a third-order derivative is:

$$\begin{aligned} f_{a,b,c} \quad = \quad & f(x+a+b+c) - \\ & f(x+a+b) - f(x+b+c) - f(x+a+c) + \\ & f(x+a) + f(x+b) + f(x+c) - f(x) \end{aligned}$$

(See Appendix A for an example of a fourth-order derivative).

So for general groups the higher-order derivatives are not the same as integrals, since in an integral one looks at the sum of all elements in a set. In groups with characteristic 2, an sth-order differential is the x-or of all 2^s different words, which means that it is also an integral.

Higher order differentials are mostly applicable to ciphers consisting of subfunctions of low algebraic degree.

To conclude this section we make some observations. In some cases an integral can contain both truncated differentials and higher-order differentials, but there are cases where integrals can be specified for more rounds than either of the other two. But in contrast to truncated and higher-order differentials, integrals do not seem to be as applicable to ciphers using non-bijective s-boxes/components, since sending a collection of texts which are all different through a non-bijective s-box does not guarantee that the output texts are all different.

5.4 The Square Attack

In *FSE'97* an integral attack was given on the cipher Square, and first known as "the Square Attack" [9]. This attack can also be used on the ciphers AES and Crypton. All three ciphers are 128-bit block ciphers operating on bytes. The sixteen bytes are arranged in a 4×4 matrix. In one round of the ciphers the following is performed on the matrix, or state:

- An addition of a sub-key
- A substitution of each byte
- A linear transformation, MixColumn, which modifies the four bytes in a column of the matrix.
- For the AES: cyclic shift of rows in the cipher

This attack was first given on Square, but due to similarities between the three ciphers the attack applied to the AES and Crypton is quite analogous. Here we will give the attack on the AES.

5.4.1 The Basic Attack

The AES operates on a 4×4 matrix of bytes called the State. It is an iterated cipher, and in each round a non-linear byte substitution (ByteSub), a cyclic shift of the rows of the State (ShiftRow), a linear mixing of the columns (MixColumn), and a round key addition. The last round does not contain the MixColumn transformation.

- The ByteSub is a non-linear byte substitution, or a bijective mapping $m : GF(2^8) \rightarrow GF(2^8)$. This mapping can be defined by a s-box which is applied to all bytes in the State.
- The ShiftRow shifts the rows in the State. The first row is not shifted, the second row is shifted once to the right, the third row is shifted twice to the right and the fourth row is shifted three times to the right.
- In MixColumn, the columns of the State are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $c(x) = 03_{16}x^3 + 01_{16}x^2 + 01_{16}x + 02$, where e.g. 03_{16} is hexadecimal.
- The round key addition is just an exclusive or of the State with the round key. The round key is derived from the cipher key by means of a key schedule.

For a complete description of the AES, see [10].

We look at an integral consisting of 256 texts, which have different values in one byte and equal values in all other bytes. This happens with the integral, round for round:

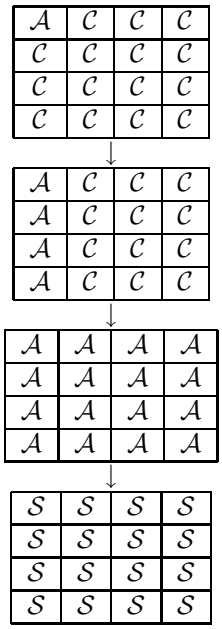

Figure 5.1: 3-round (first order) integral for the ciphers AES, Square and Cryptón, where $\mathcal{S} = 0$.

Round 1

1. The non-linear byte substitution will not affect the integral in the first round. This substitution is a bijective mapping, and following the argument in Section 5.1, this will not change the integral in the first round.
2. The ShiftRow is a cyclic shift and will not move the $\mathcal{A}$ in the first round, since the first row is not shifted in the AES.
3. The MixColumn will spread the $\mathcal{A}$ down the first column.
4. The addition of the round key does not change the integral, since this round key is constant for all words with the same position in the texts in the integral. Words that are equal before the addition will be equal after, and words that are all different will all be different after the addition.

Round 2

1. The non-linear byte substitution will not affect the integral in the second round.
2. The ShiftRow is a cyclic shift and will move the $\mathcal{A}$ so that there will be one $\mathcal{A}$ on each row and each column (on the diagonal).
3. The MixColumn will spread $\mathcal{A}$ to every entry in the matrix.
4. The round key addition will not affect the integral.

Round 3

1. The non-linear byte substitution will not affect the integral in the second round.
2. The ShiftRow cyclic shift will not affect the integral, since every entry in the matrix contains the $\mathcal{A}$.
3. The MixColumn will change the bytes, but since we have that every byte is a sum of two bytes that both takes all values in $GF(2^8)$, we have that the sum of every byte will be zero.
4. The round key addition will not affect the integral

The attack using the 3-round integral shown in Figure 5.1, on a 4-round variant of Rijndael, goes as follows:

- Choose 256 plaintexts that differ in the first byte, and obtain the corresponding ciphertexts.
- For each byte in the State
  - Repeat until the right key byte is found:
    - Guess a key byte and compute byte-wise backwards (this is possible since the last round of the AES does not contain a MixColumn transformation) to check if the sum of all 256 values is zero.

The attack on the 4 round version of the AES finds all bits of the last round key, with data $16 * 2^8$ chosen plaintexts and workload $16 * 2^8$.

5.4.2 Extension by an Additional Round at the End

This attack can be extended to an attack on a 5-round variant of Rijndael (also relevant to Square and Crypton). This is simply done by guessing 1 byte of the round key in the 5th round and additionally 4 byte of the round key in the 4th round for each key byte we want to find.

5.4.3 Extension by an Additional Round at the Beginning

We will use the same integral as before, but now we will use it from the second round and onwards. The idea is to choose a set of plaintexts that after the first round will give a collection of texts that follows the 3-round integral. We choose a collection of 2^{32} plaintexts, such that for each guess of four key bytes in the first round, one can find a collection of 256 ciphertexts after one round of encryption which form an integral going through the next three rounds just like the 3-round integral in Section 5.4.1. Figure 5.2 shows the 4-round fourth order-integral. Guess further one key byte in the sixth round and four in the fifth round, in total nine bytes. So, using the extension at the end and the beginning, we can attack 6 rounds of Rijndael using 2^{32} plaintexts, 2^{72} cipher executions and 2^{32} memory. Later this has been improved to 2^{44} by Ferguson et. al. in [12].

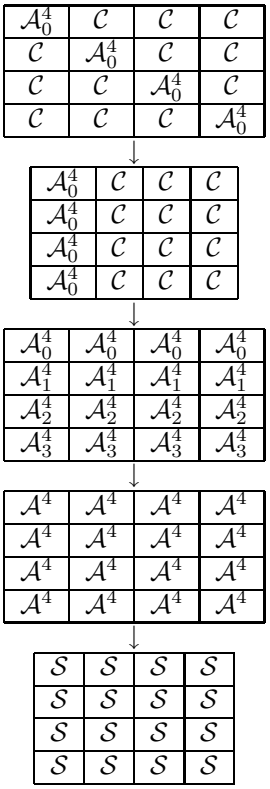

Figure 5.2: 4-round (fourth-order) integral for Rijndael.

Attack	# Plaintexts	# Cipher executions	Memory
Basic (4 rounds)	2^9	2^9	small
Extension at end	2^{11}	2^{40}	small
Extension at beginning	2^{32}	2^{40}	2^{32}
Both Extensions (6 rounds)	2^{32}	2^{44}	2^{32}

Table 5.1: Complexity of the Square attack applied to Rijndael.

So in this attack on a 128 bit block and key length version of Rijndael, we can successfully break 6 out of 10 rounds using both extensions, with less work than searching through the whole key space. This is the best attack on Rijndael known today. See table 5.1 for details of about the complexity of this attack.

5.5 Attacks on Generalized Feistel Networks

The generalized Feistel networks were proposed by Nyberg [36].

Definition 5.5.1 *[36] Let $X_0, ..., X_{2n-1}$ be the inputs to one round of the cipher. Given n s-boxes $F_0, ..., F_{n-1}$, where $F_i : \{0,1\}^d \rightarrow \{0,1\}^d$, and n round keys $K_0, ..., K_{n-1}$, the output of the round $Z_0, ..., Z_{2n-1}$ is defined as follows:*

$$Y_i = X_i \oplus F_i(K_i \oplus X_{2n-1-i}), \text{ for } i = 0, ..., n-1 \quad (5.4)$$

$$Y_i = X_i \text{ for } i = n, ..., 2n-1 \quad (5.5)$$

$$Z_i = Y_{i-1} \text{ for } i = 0, ..., 2n-1 \quad (5.6)$$

where all indices are computed modulo $2n$.

As a special case of this construction Nyberg considers the cipher where the s-boxes are bijective. For this cipher the probabilities of differentials can be upper bounded to p_{max}^{2n} where p_{max} is the maximum probability of a non-trivial differential through the s-boxes. With $n = 4$ and $d = 8$ there are s-boxes for which $p_{max} = 2^{-6}$ and the probabilities of all differentials over 12 rounds are bounded by 2^{-48}. Also, the probabilities of linear hulls over 12 rounds can be bounded by 2^{-48} [36]. This gives the cipher a good resistance against linear and differential attacks. We will later look at a generalized Feistel Network with $n = d = 8$, where the probabilities of all differentials over 12 rounds are bounded by $p_{max}^{2n} = (2^{-6})^{2*8} = 2^{-96}$.

In [23] it is shown that there exist integrals to successfully attack a generalized Feistel network assuming that the s-boxes are bijective, and that all round keys are independent and chosen uniformly at random. The attacks in [23] are given for $n = 4$ and $d = 8$. The most basic attack on this network is with an 11-round integral using only 256 texts and making it possible to attack an 12-round version of the generalized Feistel cipher by calculating backwards from the ciphertexts by guessing on key byte in the last round, and checking whether or not the first byte of all the 256 ciphertexts sum to zero. It is also possible to attack 13-round version of this cipher by guessing only three key bytes. To uniquely distinguish the right key, more than one integral may have to be used. An further extension of this attack to the 14-round and 15-round versions of this cipher can be done by similarly guessing bytes in more rounds, and checking the sum of the first byte in the output after 11 rounds. In these cases it is also necessary to repeat the attack some times to uniquely distinguish the right round-keys. The complexity of these attacks is given in Table 5.4 together with complexity of attacks from [23] on the generalized Feistel networks using higher order integrals.

A second order integral for the generalized Feistel networks with $n = 4$, $d = 8$ and bijective s-boxes is given in Table 5.3, and this integral goes two rounds further than the first order integral. The second order 13-round integral use

Ciphertexts after round								
	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
1	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
2	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
3	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
4	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
5	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$
6	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$
7	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$
8	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$
9	$\mathcal{A}$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	$\mathcal{S}$	$\mathcal{A}$
10	$\mathcal{A}$	$\mathcal{S}$	?	?	?	?	?	$\mathcal{S}$
11	$\mathcal{S}$	?	?	?	?	?	?	?

Table 5.2: An 11-round integral with 256 texts for the generalized Feistel cipher with $n = 4$, $d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$.

2^{16} chosen plaintexts, and in the trivial attack on a 14-round version of the generalized Feistel cipher using this integral ,we guess one key byte and calculate the ciphertexts after 13 rounds and check if all the 2^{16} texts sum to zero.

5.6 Attacks on MISTY

The MISTY cipher [32] was constructed by Mitsuru Matsui, and it is a block cipher with 128-bit key-size and 64-bit block-size, a Feistel structure and a variable number of rounds. MISTY is used as a generic name for MISTY1 and MISTY2, and it is designed on the basis of provable security against differential and linear cryptanalysis (see Chapter 4). Appendix B contains more details about the structures and functions in MISTY.

In [23] the best known attack today on MISTY1 is given. The idea is to use an integral where the last 32 bits of the plaintexts take all values in $GF(2^{32})$, this integral can be predicted for four rounds: $\langle \mathcal{C}, \mathcal{A} \rangle \rightarrow \langle ?, \mathcal{S} \rangle$ (see Figure 5.3). The integral is used in a chosen plaintext attack on 5 rounds of MISTY1 with data complexity 2^{34} texts and work comparable to 2^{48} trial encryptions.

This attack uses the fact that $\mathcal{S} = 0$ goes unchanged through $FL6$. To explain this, consider the input $< x, y >$ to $FL6$, the output will then be

$$\langle x \oplus [(y \oplus (x \cap KL_{61})) \cup KL_{62}] \ , \ (y \oplus (x \cap KL_{61})) \rangle$$

Since the sum in $GF(2^{16})$ is the exclusive or, we can consider this sum bit for bit, and from $\mathcal{S} = 0$ we have that $\sum x = \sum y = 0$. First consider the right part of the output $(y \oplus (x \cap KL_{61}))$. We need to show that $\sum (x \cap KL_{61}) = 0$. To see this assume that bit number i in x is '0' then so is bit number i in $(x \cap KL_{61})$. If bit number i in x is '1' then the bit number i in $(x \cap KL_{61})$ will be '0' or '1', but we know that the number of texts where bit number i in x is '1' is an even number, which means that we will have an even number of texts where the ith bit in $(x \cap KL_{61})$ is '1'. It follows that $\sum (y \oplus (x \cap KL_{61})) = 0$. We can do a similar argument for $\sum (x \oplus ((y \oplus (x \cap KL_{61})) \cup KL_{62})) = 0$.

Ciphertexts after round	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$
1	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
2	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
3	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
4	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
5	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$
6	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_1^2$	$\mathcal{A}_0^2$	$\mathcal{A}_1^2$	$\mathcal{A}_0^2$	$\mathcal{C}$
7	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_3^2$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}_3^2$	$\mathcal{A}_2^2$	$\mathcal{A}_2^2$
8	$\mathcal{A}_2^2$	$\mathcal{A}_5^2$	$\mathcal{A}_5^2$	$\mathcal{A}_3^2$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}_3^2$	$\mathcal{A}_2^2$
9	$\mathcal{A}_2^2$	$\mathcal{A}_2^2$	$\mathcal{A}^2$	$\mathcal{A}^2$	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$
10	$\mathcal{A}^2$	$\mathcal{S}$	$\mathcal{S}$	$\mathcal{S}$	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$
11	$\mathcal{A}_4^2$	$\mathcal{S}$	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$
12	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	?	?	?	?	?	$\mathcal{S}$
13	$\mathcal{S}$	?	?	?	?	?	?	?

Table 5.3: A 13-round second order integral with 2^{16} texts for the generalized Feistel cipher with $n = 4$, $d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$.

Number of rounds	Data complexity	Time complexity	Comments (all attacks from [23])
14	2^{50}	2^{50}	impossible differential
14	2^{32}	2^{32}	fourth-order integral
13	$2^{9.6}$	2^{32}	first-order 11-round integral
14	$2^{10.6}$	2^{56}	first-order 11-round integral
15	$2^{11.3}$	2^{88}	first-order 11-round integral
14	2^{16}	2^{24}	second-order 13-round integral
15	$2^{17.6}$	2^{40}	second-order 13-round integral
16	$2^{18.6}$	2^{64}	second-order 13-round integral
16	$2^{33.6}$	2^{56}	fourth-order 14-round integral
17	$2^{34.6}$	2^{80}	fourth-order 14-round integral
17	$2^{49.6}$	2^{72}	sixth-order 15-round integral

Table 5.4: Complexity of different attacks on the generalized Feistel networks from [36]. The results are from [23].

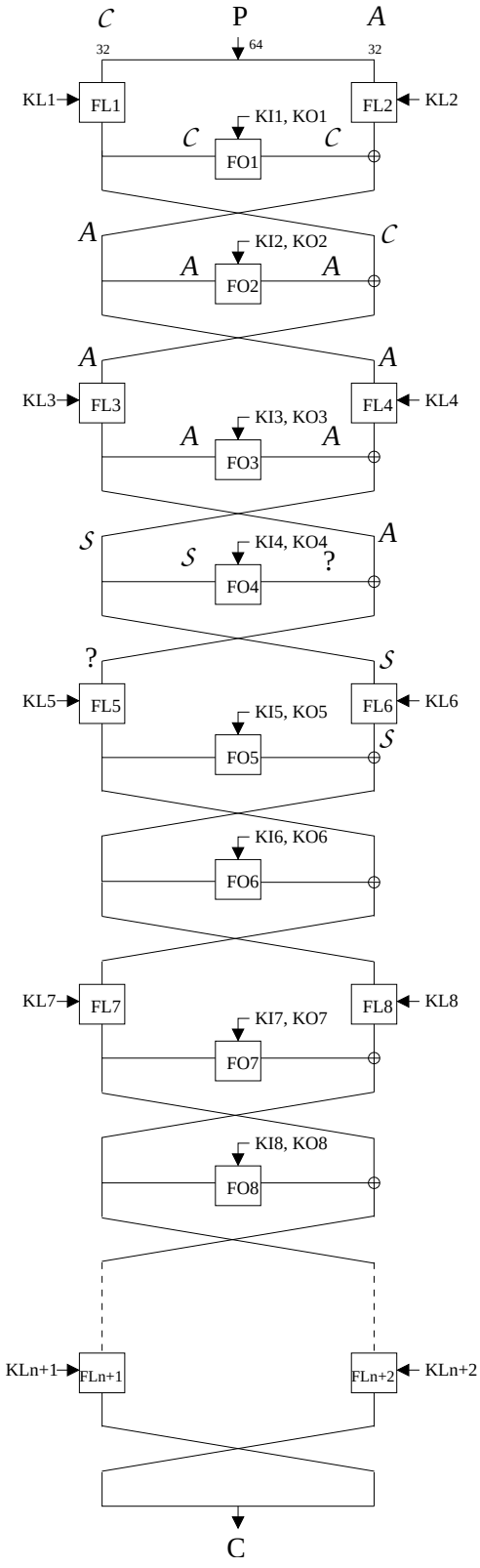

Figure 5.3: A four-round integral for MISTY1 using 2^{32} chosen plaintexts where $\mathcal{S} = 0$.

The following property of the function $FO5$ (see Appendix B) is used in the attack on MISTY1. Let $F(x, y)$ be the left half of the output of $FO5$ on the input $< x, y >$, then $F(x, y) = f(x) \oplus g(y)$, where f and g are some key-dependent bijective mappings. It was Sakurai and Zheng[40] who noted that three round of MISTY2 has this property, and $FO5$ has the same structure as 3 rounds of MISTY2. Also note that the function $FIij$ has this structure. The attack on 5 rounds of MISTY1 can be described by the following procedure:

1. Choose the plaintexts to form the integral, and obtain the ciphertexts.
2. The input to the $FO5$ function in known, and we express the left part of the output of the $FO5$ function as $f_{k_1}(left - input) \oplus g_{k_2}(right - input)$, where $k_1 = (KO_{5,1}, KI_{i,1})$ and $k_2 = (KO_{5,2}, KI_{i,2})$, totally 64 bits.
3. We now use the Sakurai-Zheng property of the $FI_{i,j}$ function, and the left output can be expressed as $f_1(left - input) \oplus f_2(right - input) \oplus KI_{i,j,1}$. The sum of the text in the integral will not depend on $KI_{i,j,1}$, and we obtain a 7-bit condition on the output from $FO5$, only dependent on the 16-bit keys $KO_{5,1}$ and $KO_{5,2}$, totally 32 bits.
4. Guess $KO_{5,1}$ and $KO_{5,2}$ and check the condition.
5. We need to repeat step 1-4 four or five times to uniquely distinguish the right candidates for $KO_{5,1}$ and $KO_{5,2}$.

Chapter 6

New Integrals

Some new integrals on the generalized Feistel networks, how to mount an attack on these networks and an integral on KASUMI will be presented in this chapter.

6.1 Generalized Feistel Networks

We have found some new integrals on the generalized Feistel networks with $n = d = 8$, which will have a comparable parameter size to the 128-bit key and 128-bit block version of the AES. This version of the AES has 10-rounds and 128-bit sub-keys or round-keys, while the generalized Feistel networks have $2 * n * d = 128$-bit block length, $n * d = 64$-bit sub-keys or round-keys. With 20 rounds the generalized Feistel networks will have a comparable performance level to the AES. Figure 6.1 shows one round of the generalized Feistel networks with $n = d = 8$.

If we can find integrals that go 19 or more rounds for this structure, this will be a good reason not to choose the generalized Feistel networks with $n = d = 8$ and bijective components as an encryption standard over the AES.

6.1.1 First Order Integral

Table 6.1 shows a new integral applying to the generalized Feistel networks with $n = d = 8$.

The integral starts with 256 plaintexts all different in only the first byte, and equal in all other bytes. Sending the integral through (5.4) does not make any change to the integral, because when we look at the first byte, which takes all 256 possible values, that byte will still take all 256 possible values after the first part of the round function 5.4, as described in Section 5.1 on page 24. As for the rest of the bytes, if the texts are equal before (5.4) or (5.5), they will be equal after (5.4) or (5.5) as well. (5.6) will only cycle the bytes one to the right. The integral will change similarly from round 1 to 8. From round 8, the byte that takes all values will start to effect other bytes. When this byte is sent through a bijective function and x-or'ed to another byte where all values are equal, the resulting byte value will also take all values. In round 12 we have one byte in the integral which takes all 256 possible values and is added to another

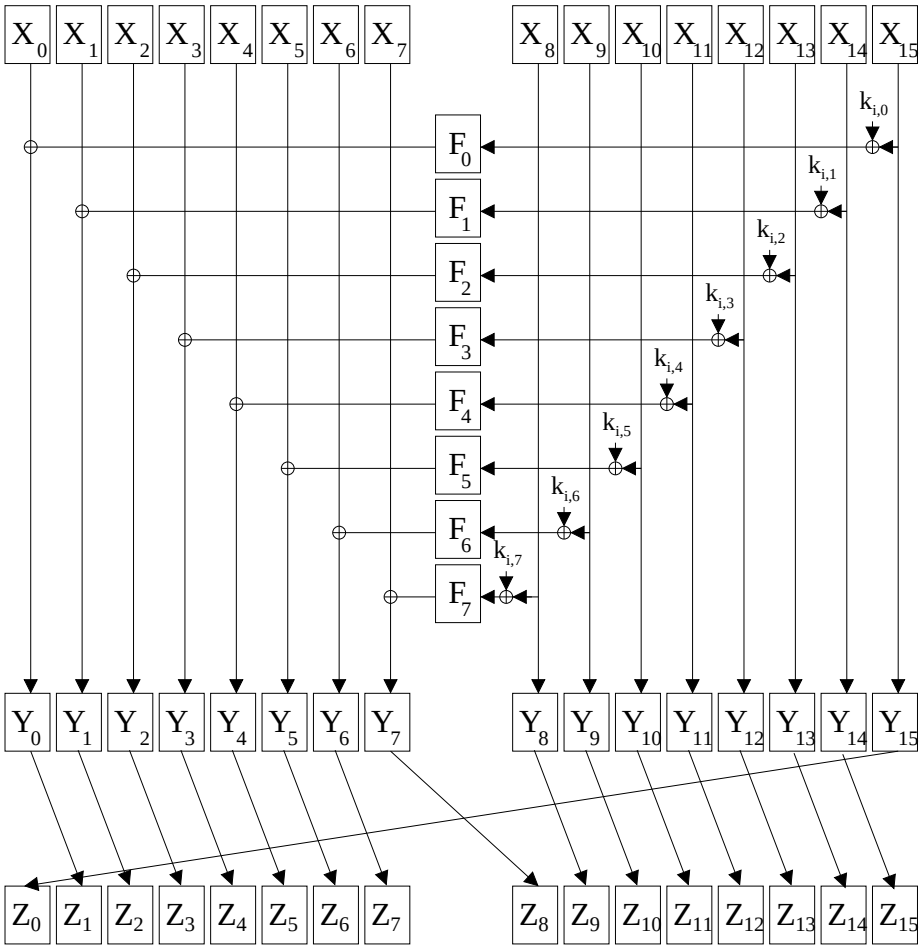

Figure 6.1: Round i of the generalized Feistel networks with $n = d = 8$, and round key k_i.

Ciphertexts after round																
	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
1	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
2	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
3	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
4	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
5	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
6	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
7	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
8	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
9	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
10	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
11	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
12	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
13	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	?	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{C}$
14	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{C}$
15	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$
16	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}$	$\mathcal{A}$
17	$\mathcal{A}$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}$
18	$\mathcal{A}$	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	?	?	?	$\mathcal{S}$
19	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 6.1: A 19-round integral with 2^8 texts for the generalized Feistel networks, with $n = d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$.

byte in the integral which also takes all 256 possible values. The sum is

$$s = \sum_{i=0}^{255} x_i + g_1(y_i),\ x_i, y_i \in GF(2^8) \tag{6.1}$$

where g_1 is a key-dependent bijective function, $+$ is the group operation in $(GF(2^8), +)$, and $x_i = x_j \Rightarrow i = j$, $y_i = y_j \Rightarrow i = j$. From 5.1.1 we known that the sum s (6.1) is 0 (zero). This $\mathcal{S} = 0$ will survive until round 19. When $\mathcal{S} = 0$ is cycled in round 19 it will end up at the beginning of the integral, and we have a distinguisher after 19 round that we can use to mount a simple attack on a 20 round version of the generalized Feistel networks with $n = d = 8$, finding 8 key-bits of the last round key.

An attack using the 19-round first order integral

1. Choose 256 plaintext, different in only the first byte, and obtain the ciphertexts from the encryption Oracle.

2. For each guess k of the 8 bit of the last round key, check:

$$\sum_{i=0}^{255} (C[i][1] + F(C[i][0] + k)) = 0 \tag{6.2}$$

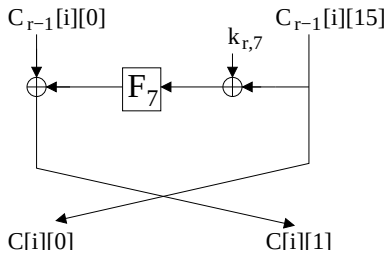

Figure 6.2: Shows how we check the condition after 19 rounds by calculating backwards from the ciphertexts after 20 rounds. C^{r-1} is the ciphertext we get when calculating one round backwards from the original ciphertext. $C[i][j]$ means ciphertexts number i, and byte number j in that ciphertext, counting from left to right.

where $C[i][0]$ is the first byte in the ith ciphertext of the 256 ciphertexts obtained in step 1, and $C[i][1]$ is the second byte in the ith ciphertext. $+$ is the group operation in $GF(2^{32})$, xor. Increase a counter for the key k if this condition is true. Figure 6.2 shows how we calculate one round backwards.

3. Repeat 1 and 2 until only one key is suggested every time, this key is the correct 8-bit part of the last round key.

Step 1 and 2 often suggest only one key, but sometimes it suggests 2 or 3 keys for one integral. Then we need to choose a different integral and repeat the key-search until only one key is suggested every time.

How many keys are suggested by each integral

If we take a closer look at (6.2) we see that it can also be written as:

$$\begin{gathered} C[0][1] \oplus C[1][1] \oplus ... \oplus C[255][1] \oplus \\ F(C[0][0] \oplus k) \oplus F(C[1][0] \oplus k) \oplus ... \oplus F(C[255][0] \oplus k) = 0 \\ \Updownarrow \\ C[0][1] \oplus C[1][1] \oplus ... \oplus C[255][1] = \\ F(C[0][0] \oplus k) \oplus F(C[1][0] \oplus k) \oplus ... \oplus F(C[255][0] \oplus k) = A \end{gathered}$$

where A is a constant for every integral if the condition is satisfied, since

$$C[0][1] \oplus C[1][1] \oplus ... \oplus C[255][1]$$

is independent of k and therefore a constant relative to each integral. So suggested keys are those who satisfy

$$F(C[0][0] \oplus k) \oplus F(C[1][0] \oplus k) \oplus ... \oplus F(C[255][0] \oplus k) = A \qquad (6.3)$$

If the 8-bit key k is not correct correct, then

$$F(C[0][0] \oplus k) \oplus F(C[1][0] \oplus k) \oplus ... \oplus F(C[255][0] \oplus k) \qquad (6.4)$$

is assumed take a random value in the range $[1, ..., 2^8 = 255]$ and it will do this for each choice of k. Since we have 255 wrong choices for k, (6.4) will take 255 random values. We only have 256 possible output values for (6.4), and for at least one of these values it is likely that the condition (6.3) is met. To sum up, the right key will always be suggested, and it is likely that 2 keys will be suggested for each integral - one right and one wrong.

Complexity of the attack

Since we only need two integrals to successfully identify the correct key, the data complexity of this attack is $2 * 2^8$. For each text in the integral we need to backtrack one round by guessing one key-byte. This gives processing complexity $2^8 * 2^8 = 2^{16}$. When testing for the key we need to keep a variable for the sum for each key, which requires 2^8 byte blocks of memory.

6.1.2 Second Order Integral

Table 6.2 shows a new integral on the generalized Feistel networks with $n = d = 8$. It is a second order integral using 2^{16} chosen plaintexts, and it starts with plaintexts where the first and the last byte together take 2^{16} different values, and the rest of the bytes are equal in all plaintexts. After 21 rounds we have a condition which we can use in an attack on a 22 round version of the generalized Feistel networks with $n = d = 8$ and bijective s-boxes.

In the first round we have the situation that the concatenation of the first byte x and the last byte y takes all values in $GF(s^{16})$. Then these bytes are x-or'ed, after that y has gone through a key dependent bijective function $g_k(y)$. We see from Table 6.2 that the string concatenation $(x \,|\, y)$ in the input to round 1 takes all values in $GF\left(2^{16}\right)$. Table 6.2 also shows that the string concatenation of the first and the second byte in the output from round one takes all values in $GF\left(2^{16}\right)$. The reason for this is that the string concatenation of the first and the second byte in the output from round one has the structure:

$$(x \oplus g_k(y) |\, y) \tag{6.5}$$

and for every fixed value of y, we have that $(x \oplus g(y))$ takes all values in $GF\left(2^8\right)$ since x takes all values in $GF(2^8)$ for every fixed value of y. (6.5) will therefore take all values in $GF\left(2^{16}\right)$. This observation also explains why this integral will go at least one round further than the first order integral. To see this observe that if we fix the second byte in the integral, the first byte will take every value, and this gives us 256 first order integrals starting after round 1 of the cipher. This is similar to the "first round trick" that is used in the integral attack on Rijndael (see Section 5.4). But, instead of just going one round further, this second order integral goes 2 round further. To explain this, we need to look at what happens to the bytes after round 9 in the integral.

Rounds 2 to 8 are similar in respect to the fact that we have a xor either between bytes that are all equal, which will give bytes that are all equal, or between bytes that are all different with bytes that are all equal, which will give bytes that are all different. After round 8 we have that the concatenation of the eighth and the ninth byte takes all values in $GF(2^{16})$.

For the rest of this section, x and y will be used to describe respectively the 9th and 10th byte in the integral after 9 rounds, and the string concatenation

$(x \mid y)$ takes all values in $GF\left(2^{16}\right)$. After 10 rounds of encryption we have the integral

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathrm{g_1(y)}, \mathrm{g_2(x)}, \mathrm{x}, \mathrm{y}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}$$

where g_1, g_2 are key-dependent bijective functions. And because for any given y, $g_2(x)$ takes all values, which gives us

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{A}_1^2, \mathcal{A}_1^2, \mathcal{A}_0^2, \mathcal{A}_0^2, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}$$

After 11 rounds we have these words in the integral

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathrm{g_5(y)}, \mathrm{g_4(x)}, \mathrm{g_1(y)} \oplus \mathrm{g_3(x)}, \mathrm{g_2(x)}, \mathrm{x}, \mathrm{y}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}$$

Since g_i are bijective functions this gives us

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{A}_3^2, \mathcal{A}_2^2, \mathcal{A}_2^2, \mathcal{A}_3^2, \mathcal{A}_0^2, \mathcal{A}_0^2, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}$$

After 12 rounds we have these words in the integral

$$\begin{gathered}\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathrm{g_9(y)}, \mathrm{g_8(x)}, \mathrm{g_5(y)} \oplus \mathrm{g_7(x)}, \\ g_4(x) \oplus g_6(g_1(y) \oplus g_3(x)), g_1(y) \oplus g_3(x), g_2(x), x, y, \mathcal{C}, \mathcal{C}, \mathcal{C}\end{gathered}$$

Which gives us

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{A}_5^2, \mathcal{A}_5^2, \mathcal{A}_3^2, \mathcal{A}_4^2, \mathcal{A}_4^2, \mathcal{A}_3^2, \mathcal{A}_0^2, \mathcal{A}_0^2, \mathcal{C}, \mathcal{C}, \mathcal{C}$$

After 13 rounds we have these words in the integral

$$\begin{gathered}\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathrm{g_9(y)}, \mathrm{g_8(x)}, \mathrm{g_9(y)} \oplus \mathrm{g_2(x)}, \mathrm{g_8(x)} \oplus \mathrm{g_1(y)} \oplus \mathrm{g_3(x)}, \\ g_5(y) \oplus g_7(x) \oplus g_{10}(g_4(x) \oplus g_6(g_1(y) \oplus g_3(x))), \\ g_4(x) \oplus g_6(g_1(y) \oplus g_3(x), g_1(y) \oplus g_3(x), g_2(x), x, y, \mathcal{C}, \mathcal{C}\end{gathered}$$

From this we get

$$\mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{C}, \mathcal{A}_6^2, \mathcal{A}_6^2, \mathcal{A}^2, \mathcal{A}^2, \mathcal{S}, \mathcal{A}_4^2, \mathcal{A}_4^2, \mathcal{A}^2, \mathcal{A}_0^2, \mathcal{A}_0^2, \mathcal{C}, \mathcal{C}$$

Here it is important to note why the 9th word in the integral is $\mathcal{S} = 0$ after 13 rounds. Expressed with our x and y, this word is:

$$(g_5(y) \oplus g_7(x)) \oplus (g_{10}(g_4(x) \oplus g_6(g_1(y) \oplus g_3(x)))) = X \oplus Y$$

where $X = g_5(y) \oplus g_7(x)$ and $Y = g_{10}(g_4(x) \oplus g_6(g_1(y) \oplus g_3(x)))$ both takes all values in $GF(2^8)$ 256 times. That is, we look at the texts where the x is fixed, and since y takes all values in $GF(2^8)$ exactly once for each x, $g_i(y)$ takes all values in $GF(2^8)$ for each $g_j(x)$. We have that for each fixed x, X and Y take all values in $GF(2^8)$ once, and since x takes all values in $GF(2^8)$ 256 times, so will X and Y. Then $\sum(X \oplus Y) = 0$.

This $\mathcal{S}$ is not changed from round 13 to round 20, and in round 21 it is cycled round and ends up in the first word in the integral after 21 rounds. Now we have a condition on ciphertexts after 21 rounds and we can mount an attack on a 22 round this version of the generalized Feistel networks.

Ciphertexts after round																	
	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$
1	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
2	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
3	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
4	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
5	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
6	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
7	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
8	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
9	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
10	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_1^2$	$\mathcal{A}_1^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
11	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_3^2$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}_3^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
12	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_5^2$	$\mathcal{A}_5^2$	$\mathcal{A}_3^2$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}_3^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
13	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_6^2$	$\mathcal{A}_6^2$	$\mathcal{A}^2$	$\mathcal{A}^2$	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$
14	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_7^2$	$\mathcal{A}_7^2$	$\mathcal{S}$	$\mathcal{S}$	$\mathcal{S}$	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$	$\mathcal{C}$
15	$\mathcal{C}$	$\mathcal{C}$	$\mathcal{A}_8^2$	$\mathcal{A}_8^2$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{C}$
16	$\mathcal{A}_0^2$	$\mathcal{A}_9^2$	$\mathcal{A}_9^2$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$	$\mathcal{A}_0^2$	$\mathcal{C}$
17	$\mathcal{A}_0^2$	$\mathcal{A}_0^2$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{A}^2$	$\mathcal{C}$
18	$\mathcal{A}^2$	$\mathcal{S}$	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	$\mathcal{C}$
19	$\mathcal{A}_4^2$	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{A}_4^2$	$\mathcal{C}$
20	$\mathcal{A}_4^2$	$\mathcal{A}_4^2$	?	?	?	?	?	?	?	?	?	?	?	?	?	$\mathcal{S}$	$\mathcal{C}$
21	$\mathcal{S}$	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

Table 6.2: A 21-round second order integral with 2^{16} texts for the generalized Feistel networks with $n = d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$.

The attack on 22 rounds of the generalized Feistel Networks

This integral can be used to attack a 22 round version of the generalized Feistel networks in almost the same way as in Section 6.1.1, but using 2^{16} chosen plaintext in each integral.

The complexity of the attack

Since we on the average only need two integrals in this attack to successfully identify the right key, the data complexity of this attack is $2 * 2^{16}$. For each text in the integral we need to backtrack one round by guessing one key-byte, this gives processing complexity $2^8 * 2^{16} = 2^{24}$. As for the first order attack, we only need a look at one ciphertext at a time, and keep track of the sum for each key we are testing, which gives memory complexity 2^8 bytes.

6.1.3 Even Higher Order Integrals

The first round trick that gave us the second order integral of the generalized Feistel networks can be extended. Every time the order is increased by two, we "move" the starting point of our collection of first order integrals. For fourth order, we fix the second and third byte after round 1 and get 2^{16} second order

Integral	Time	Data	Memory (bytes)	Rounds attacked
19 round first order	$O(2^{16})$	$O(2^8)$	2^8	20
21 round second order	$O(2^{24})$	$O(2^{16})$	2^8	22
fourth order	$O(2^{40})$	$O(2^{32})$	2^8	at least 23
sixth order	$O(2^{56})$	$O(2^{48})$	2^8	at least 24
eighth order	$O(2^{72})$	$O(2^{64})$	2^8	at least 25
tenth order	$O(2^{88})$	$O(2^{80})$	2^8	at least 26
twelfth order	$O(2^{104})$	$O(2^{96})$	2^8	at least 27
fourteenth order	$O(2^{120})$	$O(2^{112})$	2^8	at least 28

Table 6.3: Complexity of attacks on the generalized Feistel networks with bijective s-boxes and $n = d = 8$.

integrals that start after round 1. We have similar results for higher order integrals, shown in Table 6.3. A natural question would be how high the order of the integral can be, and still be faster than exhaustive search for the key. When we increase the order of the integral for the generalized Feistel networks, we have to increase it by 2 because of the way the cipher is constructed. We have given examples for first and second order integrals, and in Table 6.3 the complexity for integrals for the generalized Feistel networks with $n = d = 8$ up to fourteenth order are given. Sixteenth order integrals do not give us any information, since that will be the all possible input to the cipher, which is bijective by definition for a fixed key, which trivially gives all possible output.

6.1.4 Implementation of the Attacks

The attack on a 20 round generalized Feistel network with randomly chosen s-boxes was implemented, and we ran tests with chosen plaintexts and random keys. 8 bits of the key was recovered using one first order integral in 31 out of 100 tests. Average over 100 000 tests, it took 1.6 integrals to find the right key, and the right key was found using at most 2 integrals.

The attack on a 22 round generalized Feistel network found the right key using 1.6 integrals in average over 1000 tests, and the right key was found using at most 2 second order integrals.

We also implemented a program as a help to find and check integrals. This worked well for first and second order integrals, but to check fourth order integrals with 2^{32} texts were too time consuming. We did not implement attacks with fourth and higher order integrals for the same reasons, too high complexity to be implemented with our limited resources.

The source code for these programs can be found in Appendix C.

6.2 KASUMI

KASUMI[1] is based on MISTY, and it is the international standard encryption algorithm for Third Generation Mobile Communications Systems. Since it is based on MISTY it would be likely to believe that the attacks on MISTY would be applicable to KASUMI as well. We observe that the 4 round integral given on MISTY1 in Section 5.6, i.e. the four-round integral $\langle \mathcal{C}, \mathcal{A} \rangle \rightarrow \langle ?, \mathcal{S} \rangle$, also

is a 4-round integral for KASUMI (see Figure 6.3). This integral is actually applicable to all DES-like ciphers with only bijective components.

We have a distinguisher after 4 rounds of KASUMI and we can use this to attack 5 rounds of KASUMI. The structure of KASUMI is a bit different to MISTY1, so we cannot "copy" the attack from Knudsen and Wagner [23]. The structure of KASUMI is given in Figure 6.3, and the functions in KASUMI is given in Figure 6.4. Note that the FL and FI functions have been changed. The FI function in KASUMI has four rounds and does not have the Sakurai-Zheng property anymore.

If we just try to guess the last round key KL_5, KO_5 and KI_5 and check the 32 bit condition we get from the integral, the attack would not be faster than exhaustive search. The round key consists of 128 independent key-bits derived from the 128-bit cipher key. One refinement of this attack is to use the Sakurai-Zheng property of the FO function, which gives us that the left part of the output from FO function can be written as $f_{KO_{51}}(x) \oplus g_{KO_{52}}(y)$ where $\langle x, y \rangle$ is the input. To find the input $\langle x, y \rangle$ we need to guess the keys to the FL_5 function, since we cannot predict the sum of the outputs from FL_5 even if we know the sum of the inputs, when this is different from 0 (zero). The Sakurai-Zheng property of the FO function together with the integral give us a 16 bit condition that can be checked by guessing 96 bits of the last round key. This attack is not faster than exhaustive search either, because of two things. We need to backtrack one round for each of the 2^{32} ciphertexts in the integral, and repeat this for all guesses of the 96 bits. This alone gives time complexity 2^{128}. Since we have 2^{96} possibilities for the key, and only a 16-bit condition, each integral will suggest 2^{80} possibilities for the key. This means that we need five or six integrals to uniquely distinguish the key.

6.3 Conclusion

We have looked at the generalized Feistel networks with parameters that are comparable to those of a 128-bit block size and key size version of Rijndael, the Advanced Encryption Standard. The best known attack on the AES is an integral attack which breaks 6 out of 10 rounds using $O(2^{44})$ time, $O(2^{32})$ chosen plaintexts and $O(2^{32})$ memory. This version of the AES has comparable parameters to the generalized Feistel networks with 20 rounds and $n = d = 8$. We have shown attacks on the generalized Feistel networks that can be mounted on up to 28 rounds. This leads us to the conclusion that the generalized Feistel networks with $n = d = 8$ do not give any advantage over the construction of the AES.

From the integrals we can see that the problem with the generalized Feistel networks is that these structures have less diffusion than the AES per round. In the integral for the AES, it only takes two rounds before every byte is influenced by the first byte in the integral, while in the generalized Feistel Networks with $n = d = 8$ it takes 16 rounds. One way to fix this could be to change the round function in the generalized Feistel networks so that each byte influence more bytes per round, as in the AES function.

We observe that it is easy to use a distinguisher in an attack for both the AES and the generalized Feistel networks. In the distinguisher attacks that we have seen for the AES and the generalized Feistel networks, we need only to

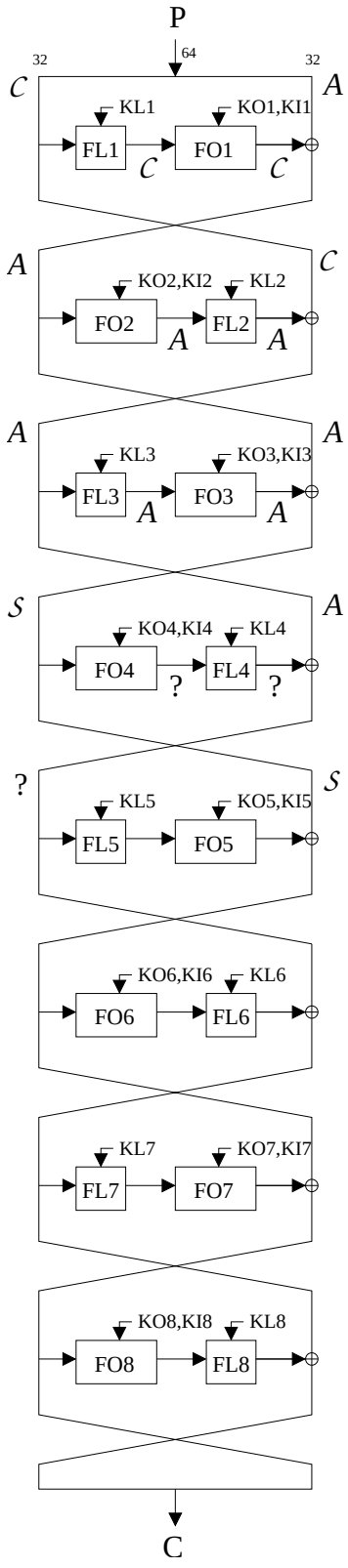

Figure 6.3: A four-round integral for KASUMI using 2^{32} chosen plaintexts where $\mathcal{S} = 0$.

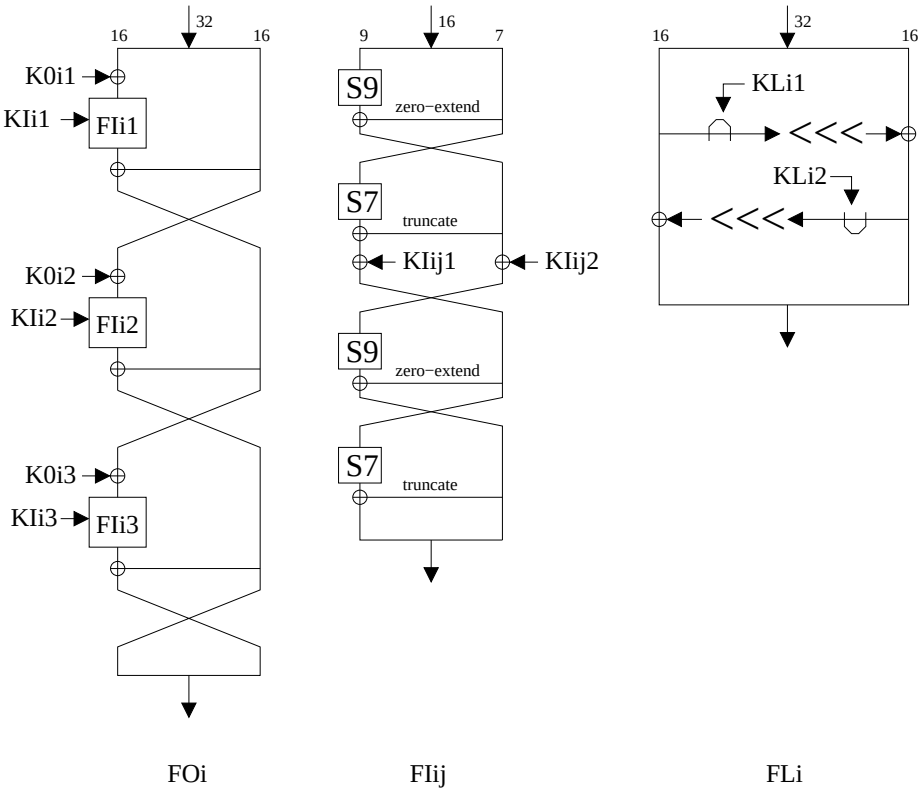

Figure 6.4: The functions in KASUMI.

guess one key byte to calculate part of the ciphertexts one round backwards and check the condition. We have also seen that this does not seem the be as easy for KASUMI, which have similarities with the generalized Feistel networks with $n = 1$ and $d = 32$, but have a round function with more complex dependency to the round key. Even though we have a distinguisher after four rounds of KASUMI it is not trivial how we can take use this in an attack on the cipher.

It is also worth noticing that the number of rounds we can predict an integral for the generalized Feistel networks seems dependent on the size of n for these networks. With $n = 1$ we have a DES-like cipher, and we can specify a first order integral that can be predicted for 4 rounds. For $n = 4$ we have a first order integral that can be predicted for 11 rounds, and for $n = 8$ we have a 19 round first order integral.

We see that if d was larger, the integral attack would require more texts, e.g. for a generalized Feistel network with $d = 16$, we would need 2^{16} texts in the first order integral. The disadvantage with a bigger d is that we need larger s-boxes. While the s-boxes in $n = d = 8$ is of size 256, the s-boxes in $d = 16$ will be of size $2^{16} = 65535$ 16 bits values, much too large for platforms with limited resources, i.e. smart cards. This can be solved by a bijective mapping $f : GF(2^{16}) \rightarrow GF(2^{16})$ defined by a function instead of a s-box, as it has been done in MISTY and KASUMI.

As others before us have remarked, it can be dangerous to focus on resistance against one attack when designing a cipher. This can make the cipher vulnerable to other attacks, as is the case for the generalized Feistel networks with $n > 1$.

There are some open question in this thesis. We do not know whether the integrals with higher order than 2 can be predicted for more rounds than what is given in Table 6.3. We have been unable to use the 4 round integral for KASUMI in a key-recovery attack, but further analysis might make it possible to find some bits of the last round using this integral with less work than exhaustive search.

Appendix A

Higher Order Derivatives

Example of a 4th-order derivative:

$$
\begin{aligned}
f_{a,b,c,d}(x) &= f_d(f_{a,b,c}(x)) = f_d(f_c(f_{a,b}(x))) = f_d(f_c(f_b(f_a(x)))) \\
&= f_d(f_c(f_b(f(x+a) - f(x)))) \\
&= f_d(f_c(f(x+a+b) - f(x+a) - f(x+b) + f(x))) \\
&= f_d(f(x+a+b+c) - f(x+a+b) - f(x+a+c) + \\
&\quad f(x+a) - f(x+b+c) + f(x+b) + f(x+c) - f(x) \\
&= f(x+a+b+c+d) - f(x+a+b+c) - f(x+a+b+d) - \\
&\quad f(x+a+c+d) - f(x+b+c+d) + \\
&\quad f(x+a+b) + f(x+a+c) + f(x+a+d) + \\
&\quad f(x+b+c) + f(x+b+d) + f(x+c+d) - \\
&\quad f(x+a) - f(x+b) - f(x+c) - f(x+d) + f(x)
\end{aligned}
$$

Appendix B

MISTY

Here we will give a short description of how MISTY works, the figures of MISTY are taken from [32]. The full details of MISTY can be found in [32].

Figure B.2 shows the data randomizing part of MISTY1 and MISTY2. We see that the 64-bit plaintext P is divided into the left 32-bit string and the right 32-bit string, which are transformed into the 64-bit ciphertext C after n rounds, using bitwise exclusive or and the sub-functions FO_i, $1 \leq i \leq n$ and FL_i, $1 \leq i \leq n+2$.

Figure B.3 shows the sub-functions FO_i,FI_{ij}, $1 \leq j \leq 3$ and FL_i. FO_i divides the input into a 16-bit left string and a 16-bit right string, which are transformed into the 32-bit output by bitwise exclusive or and the sub-functions FI_{ij}. FI_{ij} divides the input into a left 9-bit string and a right 7-bit string, which are transformed into the 16-bit output by bitwise exclusive or and by substitution tables S_7 and S_9. FL_i divides the input into the left 16-bit string and the right 16-bit string, which are transformed into the 32 bit output by bitwise exclusive or, bitwise AND and bitwise OR.

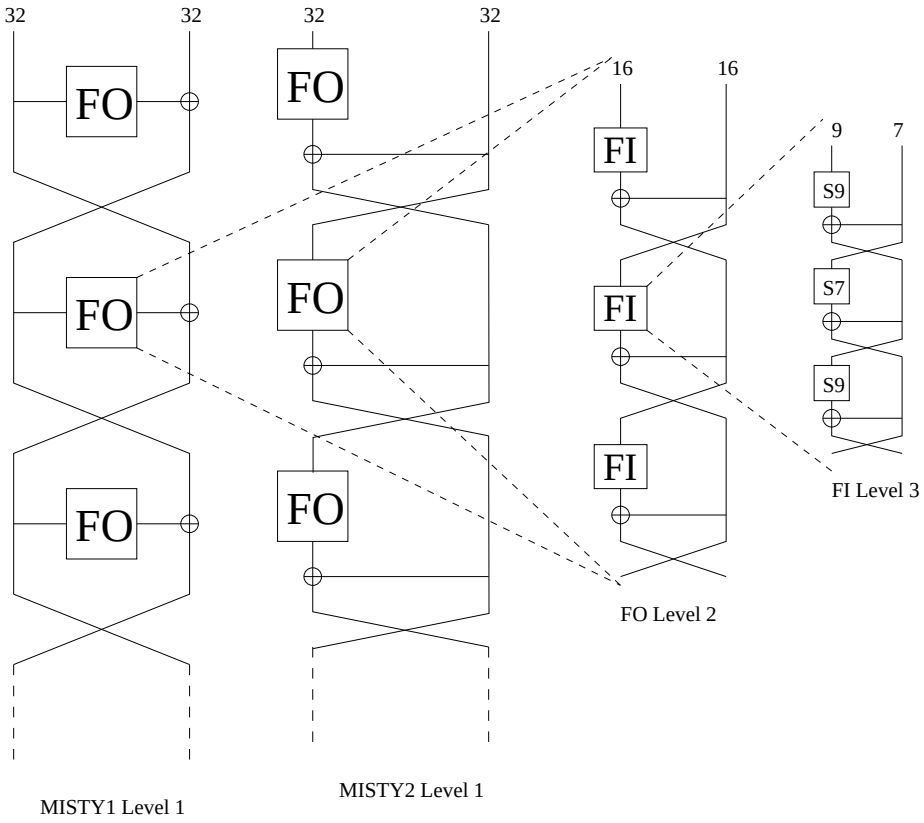

Figure B.1: Recursive structure of MISTY

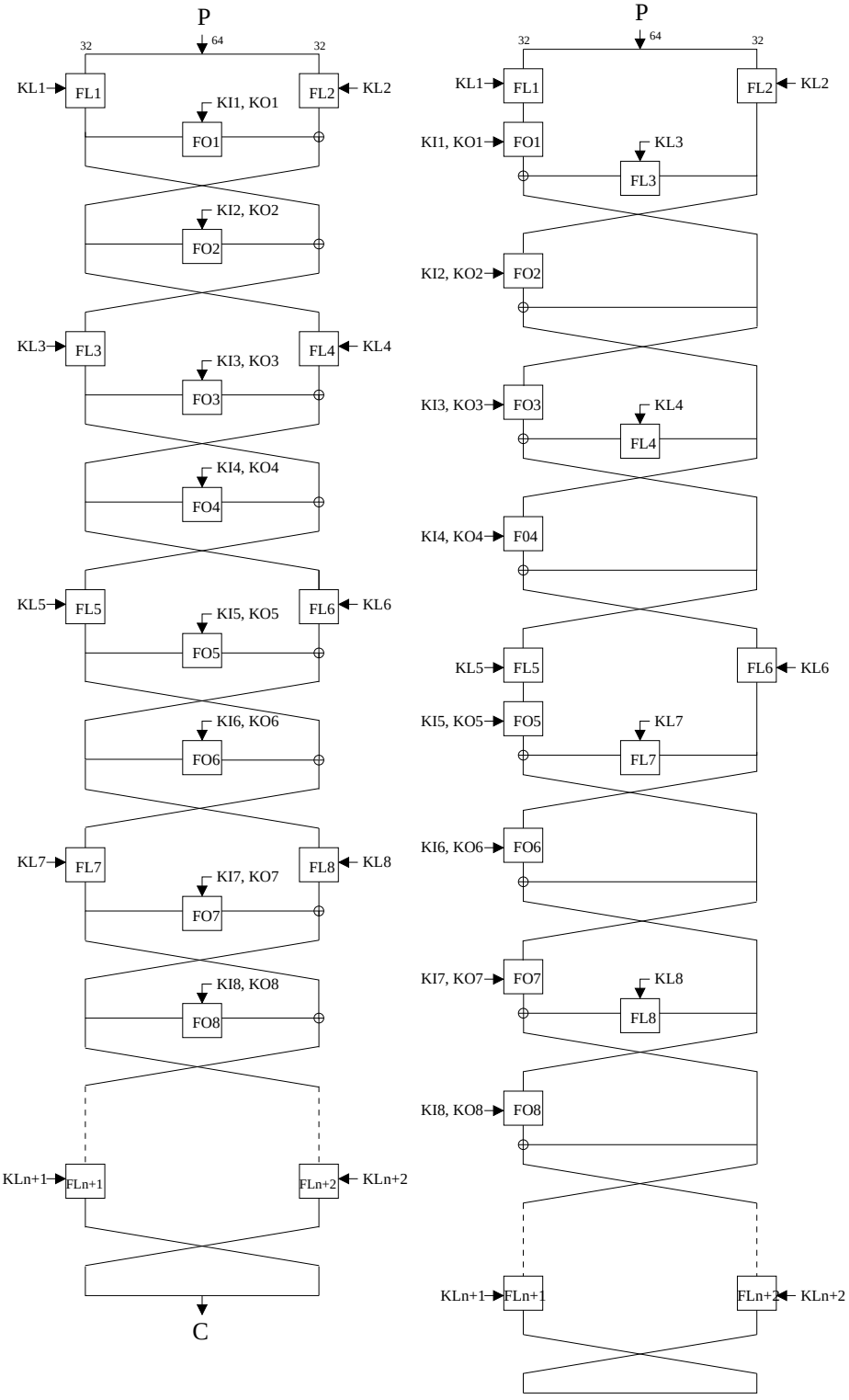

Figure B.2: The structure of MISTY1 and MISTY2

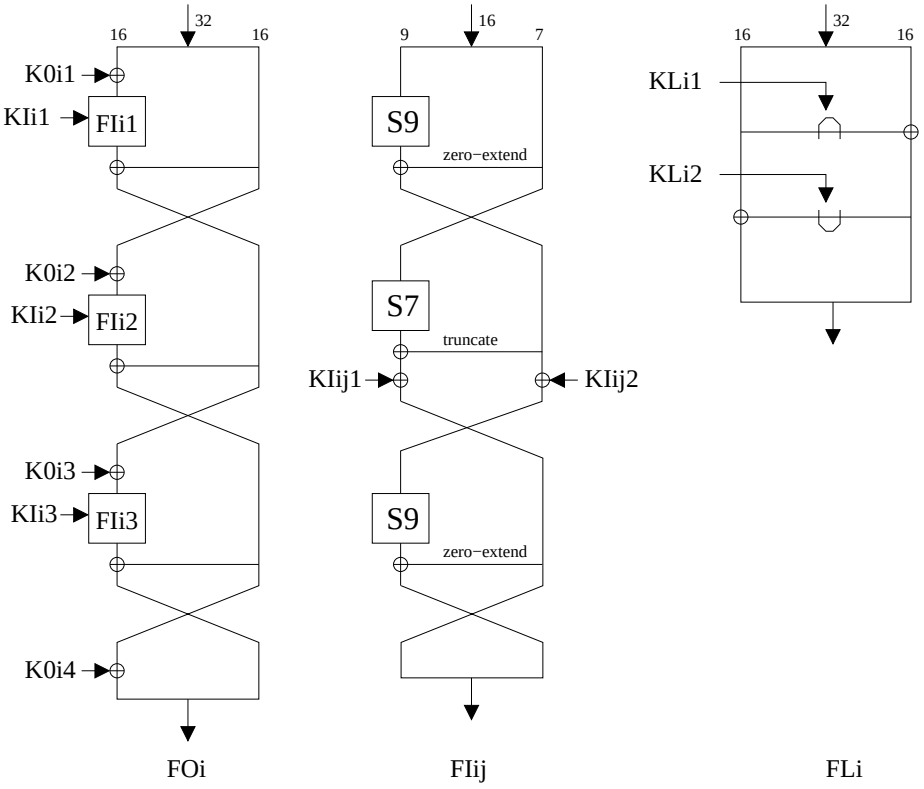

Figure B.3: The functions used in MISTY.

Appendix C

Implementations

C.1 Generalized Feistel Network

```
#include <stdio.h>
#define uint32 unsigned long
#define uint8 unsigned char
#define ROUNDS 22
#define BLOCKLENGTH 8
#define n 8
#define d 8

uint8 sbox[8][256];  /* GenFeistel s-box */
uint8 key[n];

/* Key setup. Using the same key in every round */
void key_setup(uint32 *userkey) {
  int i=0;
  for (i=0;i<n;i++)
    key[i]=(uint8)((userkey[i/4]>>(3-i%4)*8)&0xff);
}

/* Setting up sbox, using approximatly the same method as in RC4.
 * n different sboxes are made
 */
void sbox_setup(uint32 *setupKey) {
  int i=0,j=0,k=0,temp=0;
  //initialize sboxes
  for (i=0;i<n;i++) {
    for (j=0; j<256; j++) {
      sbox[i][j]=j;
    }
  }
  // mix up sboxes - only swap elements (bijection)
  for (i=0;i<n;i++) {
    for (j=0; j<256; j++) {  //j controls the step through the sbox
      k=((sbox[i][j]+(setupKey[i]>>(j%24)))%256); //update k
      //swap
      temp=sbox[i][j];
      sbox[i][j]=sbox[i][k];
      sbox[i][k]=temp;
    }
  }
}

/* Generalized Feistel encryption, assume
 * that d (from Nybergs article) is 8.
 */
void generalFeistel(uint8 *plain, uint8 *cipher) {
  int i,j,k=0;
  uint8 Y[2*n];
  // freeing plain to be changed
  for(k=0;k<2*n;k++)
    cipher[k]=plain[k];
  // the round iteration
  for (i=0; i<ROUNDS; i++) {
    //the round function, Y is a temp value
    for (j=0; j<2*n; j++) {
      if (j<n)
        Y[j]=cipher[j]^sbox[j][key[j]^cipher[2*n-1-j]];
      else
        Y[j]=cipher[j];
    }
    // Y cycled one step to the right to make next rounds ciphertext
    for(k=0;k<2*n;k++) {
      if (k>0)
        cipher[k]=Y[k-1];
      else
        cipher[k]=Y[2*n-1];
```

```
    }
  }
}
```

C.2 The Attack Using a 1. Order Integral

```
#include <stdio.h>
#include "genFeistel.c"

#define uint32 unsigned long
#define uint8 unsigned char
#define ROUNDS 20
#define BLOCKLENGTH 8
#define n 8
#define d 8
void generateIntegral();
uint8 ciphertext[256][2*n];

main() {
  int i=0,j=0,k=0,counter=0,sum=0;
  int no_suggested_keys=0,suggested_key=0, keyCounter[256], keysChecked=0, integralUsed=0;
  uint32 myKey[]={0xabababab, 0xabababab};
  for (keysChecked=0; keysChecked<100000; keysChecked++) {
    //set up new key
    for (i=0;i<1;i++)
      myKey[i]=(uint32)lrand48();
    //printf("Key to find: %.2x\n", (myKey[0]>>24)&0xff);
    for (j=0;j<256;j++)
      keyCounter[j]=0;
    key_setup(myKey);
    sbox_setup(0x7d359e1f);
    //avoid infinite loop, should exit before counter hits 10
    counter=0;
    while (counter<10) {
      counter++;
      no_suggested_keys=0;
      generateIntegral();
      for (i=0;i<256;i++) {  //step through keys
        sum=0;
        for (j=0;j<256;j++) {  //step through ciphertexts
          sum^=ciphertext[j][i]^sbox[ciphertext[j][0]^i];
        }
        // check criterium for correct key
        if (sum==0)
          keyCounter[i]++;
      }
      // find the most suggested key
      for (i=0;i<256;i++)
        if (keyCounter[i]==counter) {
          no_suggested_keys++;
          suggested_key=i;
        }
      if (no_suggested_keys==1) {
        integralUsed+=counter;
        //printf("Key found: %.2x after %i integrals\n",suggested_key,counter);
        break;
      }
    }
  }
  printf("In average was %f integrals used to find the right key (100000 tests)\n", ((double)integralUsed)/100000);
}

/*creating integral */
void generateIntegral() {
  uint8 plaintext[2*n];
  int i=0,j=0;
  for (i=0;i<2*n;i++)
    plaintext[i]=(uint8)lrand48();
  // creating the 256 ciphertexts in the 1.order integral
  for (i=0;i<256;i++) {
    plaintext[0]=i;
    //obtaining the ciphertext
    generalFeistel(plaintext, ciphertext[i]);
  }
}
```

C.3 Integral Finder

```
#include <stdio.h>
#define uint32 unsigned long
#define uint8 unsigned char
#define n 8
#define d 8
#define SIZE 256    //2^d - size of s-box
#define WORDS 16    //cipher block size in bytes
#define ORDER 2     //order of integral
#define TEXTS  65536 //number of texts in integral: SIZE^ORDER
#define ROUNDS 22   //number of rounds we want to check

uint8 plaintext[WORDS],ciphertext[WORDS]; //representing the ciphertext
```

```
uint8 sbox[n][SIZE];  /* GenFeistel s-box */
uint8 Y[WORDS]; //temp value in generalized Feistel round
uint8 key[ROUNDS][n];   //round key

void key_setup();
void sbox_setup();
void generalFeistelRound(uint8 *,int);

main() {
  int i=0, j=0, k=0, sum=0,all=0,found=0;
  uint32 counter=0;
  // we need to maintain a integral observator for each round
  uint32 integral[ROUNDS][WORDS][SIZE];
  // we also need to know the sum of each byte for each round
  uint32 number[ROUNDS][WORDS];
  //initialize structures
  for (i=0;i<ROUNDS;i++) {
    for (j=0;j<WORDS;j++) {
      number[i][j]=0;
      for (k=0;k<SIZE;k++) {
        integral[i][j][k]=0;
      }
    }
  }
  key_setup();
  sbox_setup();  //set up s-boxes
  //create the integral, handles 1.,2. and 4. order
  for (i=0;i<WORDS;i++) {
    plaintext[i]=(uint8)(lrand48()&0xf);
  }
  for (counter=0;counter<TEXTS;counter++) {
    if (ORDER==2) {
      plaintext[0]=counter & SIZE-1;
      plaintext[WORDS-1]=(counter>>8)&SIZE-1;
    }
    else if (ORDER==4) {
      plaintext[0]=counter & SIZE-1;
      plaintext[1]=(counter>>8) & SIZE-1;
      plaintext[WORDS-1]=(counter>>16) & SIZE-1;
      plaintext[WORDS-2]=(counter>>24) & SIZE-1;
    }
    else plaintext[0]=counter & SIZE-1;  //ORDER==1
    for (i=0;i<WORDS;i++) {
      ciphertext[i]=plaintext[i];
    }
    //send the ciphertext through each round and look at the result
    for (i=0;i<ROUNDS;i++) {
      generalFeistelRound(ciphertext,i);
      for (j=0;j<WORDS;j++) {
        number[i][j]^=ciphertext[j];
        integral[i][j][ciphertext[j]]++;
      }
    }
  }

  //write the integral, from round 1 to round ROUND
  for (i=0;i<ROUNDS;i++) {
    printf("After round %i:\n",i+1);
    for (j=0;j<WORDS;j++) {
      found=0;all=1;
      printf("S = %.3i, ",number[i][j]);
      for (k=0;k<SIZE;k++) {
        if (integral[i][j][k]==TEXTS) {
          printf("C");found=1;
        }
        if (integral[i][j][k]!=TEXTS/SIZE) all=0;
      }
      if (all)
        printf("A");
      else if (!found)
        printf("?");
      printf("\n");
    }
    printf("\n");

  }
}



/* Setting up sbox, inspired by RC4.
 * n s-boxes are made
 */
void sbox_setup() {
  int i=0,j=0,k=0,temp=0;
  //initialize sboxes
  for (i=0;i<n;i++) {
    for (j=0; j<SIZE; j++) {
      sbox[i][j]=j;
    }
  }
  // mix up sboxes - only swap elements (bijection)
  for (i=0;i<n;i++) {
    for (j=0; j<SIZE; j++) {  //j controls the step through the sbox
      k=sbox[i][j]^(lrand48()%SIZE); //update k
      //swap
      temp=sbox[i][j];
      sbox[i][j]=sbox[i][k];
      sbox[i][k]=temp;
```

```
    }
  }
}

void key_setup() {
  int i=0,j=0;
  //choosing a different key for every round
  for (i=0;i<ROUNDS;i++) {
    for (j=0;j<n;j++) {
      key[i][j]=lrand48()&(SIZE-1);
    }
  }
}

/* Generalized Feistel encryption, one round
 */
void generalFeistelRound(uint8 *cipher,int round) {
  int i,j,k=0;

  //the round function, Y is a temp value
  for (j=0; j<WORDS; j++) {
    if (j<n)
      Y[j]=cipher[j]^sbox[j][key[round][j]^cipher[WORDS-1-j]];
    else
      Y[j]=cipher[j];
  }
  // Y cycled one step to the right to make next rounds ciphertext
  for(k=0;k<WORDS;k++) {
    if (k>0)
      cipher[k]=Y[k-1];
    else
      cipher[k]=Y[WORDS-1];
  }
}
```

List of Figures

2.1 The secure communication between Alice and Bob, over an insecure channel, where Eve is listening in. 7

5.1 3-round (first order) integral for the ciphers AES, Square and Crypton, where $\mathcal{S} = 0$. 29
5.2 4-round (fourth-order) integral for Rijndael. 31
5.3 A four-round integral for MISTY1 using 2^{32} chosen plaintexts where $\mathcal{S} = 0$. 35

6.1 Round i of the generalized Feistel networks with $n = d = 8$, and round key k_i. 38
6.2 Shows how we check the condition after 19 rounds by calculating backwards from the ciphertexts after 20 rounds. C^{r-1} is the ciphertext we get when calculating one round backwards from the original ciphertext. $C[i][j]$ means ciphertexts number i, and byte number j in that ciphertext, counting from left to right. . . 40
6.3 A four-round integral for KASUMI using 2^{32} chosen plaintexts where $\mathcal{S} = 0$. 46
6.4 The functions in KASUMI. 47

B.1 Recursive structure of MISTY 51
B.2 The structure of MISTY1 and MISTY2 52
B.3 The functions used in MISTY. 53

List of Tables

5.1 Complexity of the Square attack applied to Rijndael. 31
5.2 An 11-round integral with 256 texts for the generalized Feistel cipher with $n = 4$, $d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$. 33
5.3 A 13-round second order integral with 2^{16} texts for the generalized Feistel cipher with $n = 4$, $d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$. 34
5.4 Complexity of different attacks on the generalized Feistel networks from [36]. The results are from [23]. 34

6.1 A 19-round integral with 2^8 texts for the generalized Feistel networks, with $n = d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$. . 39
6.2 A 21-round second order integral with 2^{16} texts for the generalized Feistel networks with $n = d = 8$ and using bijective s-boxes, where $\mathcal{S} = 0$. 43
6.3 Complexity of attacks on the generalized Feistel networks with bijective s-boxes and $n = d = 8$. 44

Bibliography

[1] 3GPP TS 35.202: "Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Algorithm Specification". http://www.3gpp.org/TB/Other/algorithms.htm

[2] K. Aoki, K. Ohta, Strict evaluation of the maximum average of differential probability and the maximum average of linear probability, preprint, February 1996.

[3] Eli Biham, Alex Biryukov, Adi Shamir. "Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials". In Jacques Stern, editor, EUROCRYPT '99, LNCS 1592, Springer Verlag, 1999.

[4] E. Biham, A. Biryukov, A. Shamir, "Miss-in-the-Middle Attacks on IDEA, Khufu and Khafre", Fast Software Encryption, LNCS 1636, L.R. Knudsen, Ed., Springer Verlag, 1999, pp. 124-138.

[5] E. Biham and A. Shamir. "Differential cryptanalysis of FEAL and N-Hash". Advances in Cryptology, Eurocrypt '91, pages 1-16, 1991.

[6] E. Biham, A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard", Springer Verlag, 1993.

[7] A. Biryukov, D. Wagner, "Slide Attacks", FSE'99, LNCS 1636, pp.245-259, Springer Verlag, 1999.

[8] P.M. Cohn. "Algebra, Volume 1". John Wiley & Sons, 1982.

[9] J. Daemen, L.R. Knudsen, and V. Rijmen, "The block cipher Square", Fast Software Encryption, LNCS 1267, Springer Verlag, 1997, pp.149-165.

[10] J. Daemen, V. Rijmen, "AES Proposal: Rijndael", AES Round 1 Technical Evaluation CD-1: Documentation, National Institute of Standards and Technology, Aug. 1998.

[11] C. D'Halluin, G.Bijnens, V. Rijmen and B. Preneel, "Attack on 6 Rounds of Crypton", Fast Software Encryption '99, Springer Verlag, 1999.

[12] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved Cryptanalysis of Rijndael", Fast Software Encryption, Springer Verlag, 2000.

[13] H. Feistel. "Cryptography and computer privacy". Scientific American, 228(5):15-23, 1973.

[14] C. Harpes, G.G. Kramer, J.L. Massey. "A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma". Advances in Cryptology - EUROCRYPT'95, LNCS 921, pp.24-38, Springer Verlag, 1995.

[15] M.E. Hellman. "A cryptanalytic time-memory trade off. IEEE Transactions on Information Theory", IT-26:401–406, 1980.

[16] I.N. Herstein, "Topics in Algebra", 2nd ed., John Wiley & Sons, 1975.

[17] T. Jakobsen, L.R. Knudsen. "The Interpolation Attack on Block Ciphers", Fast Software Encryption, Springer Verlag, 1997, pp. 28-40.

[18] D. Kahn. "The Codebreakers". MacMillan, 1967.

[19] L.R. Knudsen. "Block Ciphers - Analysis, Design and Applications". PhD thesis, Aarhus University, July 1994.

[20] L.R. Knudsen and T.A. Berson, "Truncated differentials of SAFER", Fast Software Encryption, Springer Verlag, 1997, pp.15-26.

[21] L.R. Knudsen, "Truncated and Higher Order Differentials", Fast Software Encryption, Springer Verlag, 1995, pp. 196-211.

[22] L.R. Knudsen, "A Detailed Analysis of SAFER K", Journal of Cryptology, vol.13, no.4, Springer Verlag, 2000, pp.417-436.

[23] L.R. Knudsen, D. Wagner, "Integral Cryptanalysis", Fast Software Encryption, Springer Verlag, 2002.

[24] X. Lai, "On the Design and Security of Block Ciphers". PhD thesis, ETH, Zürich, Switzerland, 1992.

[25] X. Lai, "Higher Order Derivations and Differential Cryptanalysis", Communications and Cryptography: Two Sides of One Tapestry, Kluwer Academic Publishers, 1994, pp.227-233.

[26] X. Lai, J.L. Massey, and S. Murphy. "Markov ciphers and differential cryptanalysis". In D.W. Davies, editor, Advances in Cryptology - Proc. EUROCRYPT'91, LNCS 547, pages 17-38. Springer Verlag, 1992.

[27] A.K. Lenstra, E.R. Verheul. "Selecting Cryptographic Key Sizes", Journal of Cryptology, vol.14, no.4, pp.255-293, Springer Verlag, 2001.

[28] J.L. Massey. "Cryptography: Fundamentals and applications". Copies of transparancies, Advanced Technology Seminars, 1993.

[29] J.L. Massey. "SAFER K-64: A byte-oriented block-ciphering algorithm" FSE-94, LNCS 809, pp.1-17 Springer Verlag, 1994.

[30] J.L. Massey. "SAFER K-64: One year later", FSE-95, LNCS 1008, pp.212-241, Springer Verlag, 1995.

[31] M. Matsui. "Linear cryptanalysis method for DES cipher". In T. Helleseth, editor, Advances in Cryptology - Proc. EUROCRYPT'93, LNCS 765, pages 386-397. Springer Verlag, 1993.

[32] M. Matsui. "New Block Encryption Algorithm MISTY". Fast Software Encryption, LNCS 1267, pp. 54-68, Springer Verlag, 1997.

[33] A. Menezes, P. van Oorschot, S. Vanstone. "Handbook of Applied Cryptography", CRC Press 1996.

[34] National Bureau of Standards. "Data encryption standard. Federal Information Processing Standard (FIPS), Publication 46". National Bureau of Standards, U.S. Department of Commerce, Washington D.C., January 1977.

[35] K. Nyberg. "Differentially uniform mappings for cryptography". In T. Helleseth, editor, Advances in Cryptology - Proc. EUROCRYPT'93, LNCS 765, pages 55-64. Springer Verlag, 1993.

[36] K. Nyberg. "Generalized Feistel networks", In K. Kim and T. Matsumoto, editors, Advances in Cryptology - ASIACRYPT'96, LNCS 1163, pp. 91-104, Springer Verlag, 1996.

[37] K. Nyberg. "Linear approximations of block ciphers". In A. De Santis, editor, Advances in Cryptology - Proc. EUROCRYPT'94, LNCS 950, pp. 439-444. Springer Verlag, 1994.

[38] K. Nyberg and L.R. Knudsen. "Provable security against differential cryptanalysis". In E.F. Brickell, editor, Advances in Cryptology - Proc. CRYPTO'92, LNCS 740, pages 566-574. Springer Verlag, 1993.

[39] R. Rivest, A. Shamir and L. Adlemann. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems." Communications of the ACM, 21: 120-126. February 1978.

[40] K. Sakurai and Y. Zheng. "On Non-Pseudorandomness from Block Ciphers with provable Immunity against Linear Cryptanalysis" IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Vol. E80-A, No.1, pp. 19-24, 1997.

[41] C.E. Shannon. "Communication theory of secrecy systems". Bell System Technical Journal, 28:656-715, 1949.

[42] D. Stinson, Cryptography - Theory and Practise, CRC press 1995.

[43] D. Wagner, "The Boomerang Attack", Fast Software Encryption, Springer Verlag, 1999, pp. 156-170.