Abstract

As we are entering the third generation of mobile technology (3G) the number of services needing security grows larger. To assess if the security provided by 3G is sufficient, we take a closer look at the security mechanisms and their building blocks. Within the 3G security environment the Kasumi block cipher plays an important role in the integrity and confidentiality provided. Thus the security of Kasumi, the integrity mode ($f9$) and confidentiality mode ($f8$) is vital.

In this thesis the published attacks, the provable security and the pseudo-randomness of Kasumi are examined in order to consider whether Kasumi, $f8$ and $f9$ are secure. Also described are the authentication and key agreement example set *Milenage* and it's kernel function Rijndael (AES). In addition, background theory is provided for the reader to better understand the proofs and cryptanalytic techniques used.

The results of this thesis show no threat to the security of the Milenage, $f8$ or $f9$ functions. The security of Kasumi is also preserved as it is a family of pseudo-random permutations and provable secure against linear and differential cryptanalysis.

Keywords: block cipher, provable security, pseudo-randomness, Kasumi, UMTS, $f8$, $f9$, Milenage

Foreword

This thesis was written in the period between February 2003 and July 2004 by Tor-Erik Mathisen. The thesis is the result of the final assignment leading to the Cand. scient degree at the University of Oslo, Department of Informatics.

This thesis assumes that the reader has good knowledge of computer science equal to a bachelors degree and some knowledge of basic cryptology.

The report is divided into two main parts. Part I will give a thorough background theory for the reader to understand Part II. Part II will give a state-of-the-art review of the block ciphers in UMTS(with emphasize on Kasumi), the security they provide and the environment that they are used in.

I would also like to give my gratitude to my tutor Leif Nilsen for continuous good reviews and help throughout the time spent on this thesis. And to Niels Petter Callin and Jan-Walter Parr for many helpful hints.

Tor-Erik Mathisen

Kjeller, August 2004

Contents

Introduction

The world today has become a place were "everybody" needs to be available anywhere, anytime. Available to receive a phone call, read e-mail or get the latest news anywhere you like. Availability through portable solutions has become a large business, and thus many make a profit from it. However, availability and portability has a prize. With wireless networks and mobile phones a whole new set of intrusions may be possible. It has therefore become more important to protect connection points, the transmitted data, and the terminals.

Different technologies have different solutions, with different degrees of success. Within the mobile phone technology we have reached the third generation. Each generation has learned from the previous, and hopefully the developers have been able to create a technology that can keep the user available while simultaneously protecting the user information, the data, and the radio network.

During the last few years we have also seen a major development within the area of cryptography. New strong ciphers have been developed with immunity against several cryptanalytic attacks. However, new attacks have been made, stronger and more effective than the old ones.

Problem description

To be able to assess whether or not the third generation of mobile technology (3G) keeps it's promise or not (secure for the next 20 years), I wanted to take a closer look at the technology of 3G, especially at the security mechanisms and features.

I wanted to describe how block ciphers where used within UMTS and see what kinds of proofs of security and possible threats that existed. Thereby verifying whether the security mechanisms (and features) of 3G/UMTS provided sufficient security, and whether there were any weaknesses/attacks that compromised this security. Thus, investigate what the state-of-the-art

research within the field of cryptology could tell us about the security of the block ciphers, operation modes and other security functions of the UMTS.

In order for the reader to appreciate this security analysis, I have incorporated enough background information to make the theory of provable security more accessible to the reader.

Thus, this thesis describes and assesses the encryption, integrity and authentication of UMTS, and in addition brings the reader up-to-date within the best proofs of block cipher security that today's cryptology science can provide.

Thesis justification

At the time when I started this thesis (spring 2003), no complete or even half complete security assessment of the security mechanisms of UMTS and their building blocks existed. Several individual papers describing different attacks and the assessment of individual UMTS security functions did exist, but no complete guide.

In addition, proving the security of block ciphers is a relatively new area of science, and thus notations, expressions etc. may vary from paper to paper. No complete background theory for this science seemed to have been gathered - it was just assumed that everyone knew it.

Thus, the need for a state-of-the-art overview with a thorough background theory was imminent both for the UMTS security functions and the provable security of block ciphers (Kasumi).

It is my hope that this thesis can help cryptographic beginners to get a better and easier start learning the topics of block ciphers, their security, and how to use the security in a larger scale, such as the UMTS environment.

Thesis outline

Chapter 1 trough 6 are "background theory" for the reader to understand chapter 7 trough 10.

Chapter 1 - Introduction to cryptography Contains an introduction to the art of cryptography, where the basic vocabulary of cryptography is discusses.

Chapter 2 - Block ciphers Describes and defines block ciphers, their design and security.

Chapter 3 - Pseudo-random functions and permutations Gives a thorough background theory of pseudo-random functions and permutations and their role within the art of cryptography.

Chapter 4 - Operation Modes This chapter gives an overview the operation modes most relevant to the operations modes of UMTS.

Chapter 5 - Introduction to cryptanalysis Chapter 5 gives a introduction to cryptanalysis with an overview of the most common attacks (and most relevant) on block ciphers.

Chapter 6 - UMTS - Universal Mobile Telecommunications System The UMTS chapter describes the architecture and security architecture of UMTS. It also describes the different security mechanisms and features needed within the 3G environment.

Chapter 7 - Kasumi Describes the Kasumi block cipher, all the way from the design principles to implementations aspects.

Chapter 8 - Security of Kasumi and the $f8$ and $f9$ functions This chapter analyzes and describes how to prove Kasumi immune to some types of cryptanalysis and gives a description of the best possible attacks published against Kasumi existing today.

Chapter 9- Rijndael - AES Gives a small description of the new Advanced Encryption Standard (AES), and tries to explain why it was chosen as a kernel function for the Milenage example set.

Chapter 10 - Milenage Milenage is the authentication and key agreement functions example set provided by the 3GPP. This chapter will describe the functions and give an overview of possible attacks that are published.

Chapter 11 - Thesis conclusion This chapter is dedicated to give a summary and a conclusion of the whole thesis.

Chapter 12 - Further work Chapter 12 presents recommendations for further work, such as for a Ph.D or in an another masters thesis.

Part I

Background material

Chapter 1

Introduction to cryptography

Cryptology is the collective term of *cryptography* and *cryptanalysis* (see chapter 5 for the introduction to cryptanalysis). Cryptology is an old art running all the way from the ancient Egyptians 4000 years ago an up until today[1]. Humans have always had, and will always have, an urge to keep important information to them self and only available to those intended. Cryptology has also played an important role in many wars, and has often swayed the outcome into the direction to the side with the most clever cryptographers or cryptanalysts. Knowledge and the use of cryptology will probably play an equally (or probably more) important role in the future as it has in the past.

The practitioners of the two terms, cryptography and cryptanalysis, are in a constant struggle to get a lead on each other. Cryptographers shall provide confidentiality, integrity and authentication between the authorized entities, while the cryptanalysts try to break down what the cryptographers have built up. However, at the same time it is the two struggling forces that drive each other and thus cryptology forward; one branch obsolete without the other.

This chapter will give the reader a short introduction to cryptography and the basic concepts and terms. In addition we will see how the *Symmetric-key trust model* is used to provide confidentiality, integrity and authentication.

1.1 Cryptography

As mentioned cryptography is the study of techniques for providing the authorized entities with confidentiality, integrity and authentication in and around their information flow through insecure channels[1]. The entities are often two parties A (Alice) and B (Bob) that want to "speak" to each other

without any other O (Oscar, the opponent/adversary) entities hearing the words (confidentiality), impersonating as A/B (authentication) or changing the contents of the information (integrity).

There are two models or protocols that try to provide confidentiality, integrity and authentication between two entities, and these are called *Symmetric-key trust model* and *Asymmetric-key trust model* (Public-key model)[2]. Only the first one will be discussed here, due to the fact that it is just this model that is used in the world of UMTS and in this thesis. However, [2] contains information about asymmetric-key trust model for the curious reader.

My description of the Symmetric-key trust model is mainly based on [2] and [1].

1.2 Symmetric-key trust model (STM)

STM is a model where the entities (Alice and Bob) share a secret key unknown to the adversary (Oscar), thus the name *symmetric*. It is the knowledge of this key that allows Alice and Bob to transmit their information securely.

The security of STM relies fully on the secrecy of the key K. The key is a string of bits that we *trust* is unknown to the adversary. However, the STM does not concern itself with how the entities got the key, only that they have it.

To provide the entities with the goals of cryptography (confidentiality, integrity and authentication) two main protocols or schemes are available within the Symmetric-key trust model. To provide confidentiality we have the *Symmetric-key encryption scheme*, and for integrity and authentication we have the *Message authentication scheme*.

1.2.1 Symmetric-key encryption scheme (SES)

The symmetric-key encryption scheme (see figure 1.1) is a method or protocol for providing confidentiality, by encryption, between Alice and Bob.

SES consists of three rules/algorithms (E, D, K). E is the encryption rule, D is the decryption rule and K is the key that is shared between Alice and Bob. E and D uses the key K in order to encrypt/decrypt. E and D is known to all parties, Alice, Bob and Oscar, but the key is known only to Alice and Bob.

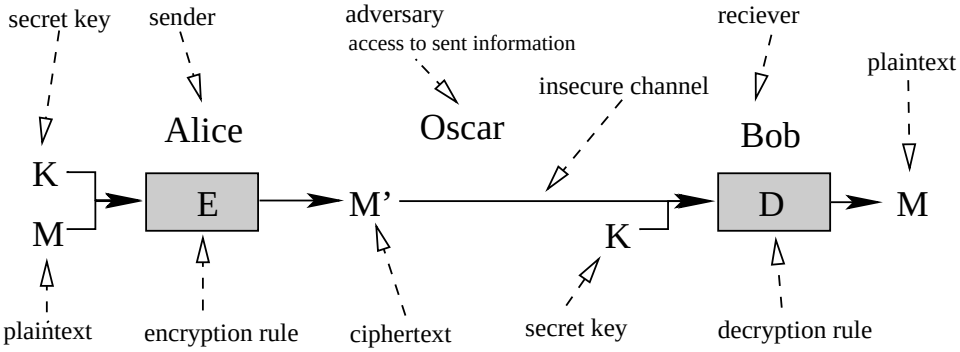

Figure 1.1: Symmetric-key encryption scheme

1.2.1.1 Functional description:

Alice wants to send a secret message M (plaintext) to Bob. She sends the plaintext and her secret key K shared with Bob as input to the encryption rule E. As output from E comes the encrypted message M' (ciphertext). The ciphertext is then transmitted over an insecure channel (such as the internet) to Bob. When Bob receives the message he applies the decryption rule D with the ciphertext from Alice and his version of the key K as input, and gets the plaintext M as output[1]

1.2.1.2 Encryption/decryption algorithms:

In the symmetric-key encryption scheme the encryption/decryption algorithm can be implemented as two types of algorithms/ciphers. The two are *block* and *steam* ciphers, where the block cipher is most common. The encryption algorithms used in UMTS (Kasumi and AES, chapter 7 and 9 respectively) are both block ciphers.

Block cipher (bc): The bc takes a block of plaintext bits and a key as input and returns a block of ciphertext bits (block ciphers are described in chapter 2). However, to become a adequate encryption rule the bc needs a *operation mode* (see chapter 4 for more information) to encrypt more than one block.

Stream cipher: The stream cipher creates a continues stream of bits, with length equal to the plaintext, that is xored ($\oplus$) with the plaintext to create the ciphertext.

[1]The decryption rule D is often implemented as E^{-1}

1.2.2 Message authentication scheme/code (MAC)

With the MAC scheme we want to accomplish two things; authentication and integrity. I.e. we want to know that the message sent is not tampered with, and that the message originates from the acclaimed entity. In order to accomplish this the scheme consists of three rules/algorithms (K, T, V), where the K is the shared key, T is the one-way function and V is the verification rule.

1.2.2.1 Functional description:

Alice wants to send her message M to Bob, but first she sends M and the secret shared key K as input to the one-way function T creating the "fingerprint" P, $P = T(M, K)$. Then she sends (M,P) to Bob.

Bob, upon receiving (M,P), sends M, P and his K to the verification rule V, $true/false = V(P, M, K)$. If V returns $true$ or 1, M and P matches and he know that M is not tampered with and that the message originates from Alice, due to the fact that only Alice(or those with knowledge of the key K) can make the fingerprint P. If V returns $false$ or 0 the fingerprint does not match, Bob knows that either M or P is tampered with.

1.2.2.2 MAC algorithms:

To create this MAC or fingerprint special functions are designed that take any string (with any length) as input and creates output with a fixed output length. Thus, it is an infinite number of plaintexts per output. It is therefore vital to makes sure that it is infeasible to forge or modify the output.

1.2.3 Remarks

An important word here is *trust*. Often the key is selected from a large set, say with 2^l different keys, and it is therefore possible for the adversary to guess the right key. Typically $l = 128$, in this case the probability p for a correct guess is small $p = 2^{-128}$, but existing.

The length n of the plaintext is preserved when converted into ciphertext. It is therefore possible to guess the correct plaintext by randomly selecting n bit. Here the probability is 2^{-n}.

The important lesson here is that the upper bound for the security and trust is defined by probability.

1.3 Summary

The symmetric-key trust model provides the entities with confidentiality (symmetric-key encryption scheme), authentication and integrity (message authentication scheme). The security of these schemes lies with the secrecy of the shared key.

This chapter has given the reader an introduction to symmetric-key encryption and some basic cryptography terms uses throughout this report. The notions and concepts of cryptology used in this report grows rapidly and it is therefore vital to get the basic understanding of how cryptography is applied to the message flow.

Chapter 2

Block ciphers

If you look up the two words *block* and *cipher* in the dictionary they will both have several meanings. *Block* may mean "a collection of objects seated near each other", and *cipher* can mean "a message written in a secret code" or "make a mathematical calculation or computation". Combined, these three definitions give a good description of what block ciphers really are. A possible definition of a *block cipher* can be "a collection of bits written in secret code created by a mathematical calculation."

In the symmetric-key encryption model the block cipher is used as a tool to provide confidentiality between two or more parties, and is currently the most popular way[2]. However, a block cipher in it self does not provide any confidentiality. The cipher has no practical use without the assistance of *operation modes* (See chapter 4). It is also vital that the secret key, used for encryption, remains secret and that the *mode of operation* used, does not contain any known weaknesses.

To better understand what block ciphers are and how they are designed, the remainder of this chapter will focus on design principles, requirements and "best practices" for block ciphers.

2.1 Definitions of block ciphers

The mathematical definition of a block cipher is that the block cipher is a function E, defined in equation 2.1:

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \tag{2.1}$$

that takes two inputs, key K and plaintext P. K having length k and P having length n. The output from E is the ciphertext C, also with length n.

In other words the block cipher is a set of many functions [1] mapping blocks of plaintext to blocks of ciphertext. When deciding for a key, what you really do is selecting one of the 2^k different mappings/functions. Each of these mappings is one-to-one, i.e. only one plaintext gives a specific ciphertext for the chosen key and visa versa. This one mapping is often denoted $E_K(P) = C$, whereas the mathematical notation is given in equation 2.2. By definition the inverse function E_K^{-1} does also exist, and is used when decrypting.

$$E_K : \{0,1\}^n \rightarrow \{0,1\}^n \tag{2.2}$$

This means that E_K is a permutation (See section 3.1.1 for more information) on the plaintext block. This permutation, E_K is one of $2^n!$ different possible mappings of a block of plaintext to a block of ciphertext.

To calculate this number we know that the plaintext space contains 2^n different blocks. Each block can correspond to one and only one block of the ciphertext space. For the first plaintext block you can choose between 2^n cipher blocks, for the second $2^n - 1$, for the third $2^n - 2$, ..., for the last 1 block, resulting in equation 2.3

$$2^n! = (2^n) * (2^n - 1) * (2^n - 1) * ... * 1 \tag{2.3}$$

2.1.1 A block ciphers goal

The number $2^n!$ is vast and out of bounds for today's polynomially bound computers, even for $n \geq 6$. The ultimate goal for a block cipher is that it may be able to blend in amongst the $2^n!$ mappings and thereby become undistinguishable from a random chosen mapping, from all the $2^n!$ possible mappings. A block cipher with this property is called a *pseudo random permutation* and is covered in section 3.

2.2 Security of block ciphers

2.2.1 Security levels

When discussing the security of block ciphers three levels are often defined. They are used when classifying the level of security that a cipher provides.

A cipher evaluated to be in the *provable security* level is also computational secure, but not visa versa. Each new level is a step up the security ladder.

As mentioned above, if the goal of a block cipher is to be undistinguishable from a random chosen permutation, the ultimate goal is to be unconditional

[1] The number is 2^k, equal to the number of different keys

secure. However, the ultimate goal is not (yet) practical, see example of the one-time-pad in section 2.3.

computational security If a cipher is computational secure the amount of computations required to break the cipher is larger than a large integer N. All possible known attacks are considered, but computational security gives no guarantees against unknown attacks. This N is determined by how fast modern day computers can process information. DES was previously considered computationally secure with a key space of 2^{56} different keys, but today's computers have made DES insecure.

provable security When a cipher is provable secure the breaking of the cipher is equal to breaking a known hard problem. The security of the cipher is linked to the given problem and is no harder than the problem. Solve the problem and the cipher is broken.

unconditional security A unconditional secure cipher cannot be broken. Even if the adversary is given unlimited computational power, space and/or time.

2.2.2 Security requirements

For a block cipher even to be considered secure some basic requirements must be fulfilled. The first and perhaps the most obvious is the size of the key.

2.2.2.1 Key size

The key size, k, must at least long enough such that it is impossible, with today's computers, to do an exhaustive search through all 2^k different keys. The reason for this is that the key size is the upper bound for the complexity of an attack on the block cipher.

The key size should also be the lower bound for the security, i.e. no attack exists that is better than exhaustive search. If such attack should exist, we say that the cipher is *broken*, or that the attack was able to *break* the cipher.

On the other hand, the key must not be too large either, increasing the complexity of the cipher and the key handling, ultimately decreasing the speed of the cipher. A key size of $k \geq 128$ bits is considered enough.

Example: AES has a key size of 128, 192 or 256 bits. Kasumi has a key size of 128 bits.

2.2.2.2 Block size

The block size is the amount of bits (input) that the block cipher can process in one computation. Some of the same considerations as for key size also apply here. The block size must not be to small, the reason for this is that the statistical dependencies, between plain- and ciphertext, will shine through.

A too large block size will make it harder to mix all bits in the plaintext, and thereby decreasing the encryption/decryption speed.

A block size of either 64 or 128 bits is the norm today.

Example: AES has a block size of 128 bits. It was expressed wishes to also include the block sizes 192 and 256 to AES, but the wish was not granted[3].

Kasumi has a block size of 64 bit.

2.3 Design principles

The perfect block cipher with unconditional security does exist, having the property that no statistical relations exist between the cipher- and the plaintext. Then it is not possible to know any more about the plaintext after you have seen the ciphertext, than before. This definition was given by Shannon in [4]. Some ciphers fulfill this property, but they all require an unreasonable large key.

> **Example - The one-time-pad:** The one-time-pad is an unconditional secure cipher, where the key size is as long as the plaintext. The ciphertext is the xor product of the key and the plaintext. In addition to the impractical size of the key, each key is only usable once, due to the fact that $(C \oplus C') = (K \oplus P) \oplus (K \oplus P') = P \oplus P'$.

Instead of searching for unconditional secure ciphers the scientists searched for the best possible approximation. Combined with that the design of block ciphers also have been strongly influenced by the types of attacks that existed, the scientists tried to find a block cipher secure against all known attacks. They were less interested in finding a cipher fulfilling the one notion of security, that guaranteed for the confidentiality of the information, and more in finding a computational secure cipher that could be implemented in practice.

With Shannon's theory in mind, the notions of confusion and diffusion where adopted in the design of many block ciphers.

2.3.1 Confusion and Diffusion

Confusion is the principle of hiding the statistical relations between the plaintext, key and the ciphertext. This is often accomplished by the use of substitutions.

Diffusion is the principle of that every bit of the ciphertext should depend on every bit of the plaintext and the key. A change in one bit of the plaintext should result in a change of approximately 50% of the ciphertext bits. The tool for applying diffusion to the cipher is permutations (xoring with a subkey), rearranging the bit order.

Normally one need to apply more than one permutation and substitution to establish the satisfactory level of security (computational secure). The most common procedure is to create a *round function* consisting of one permutation and one substitution, and then apply this function several times. Each round has it's own *subkey/roundkey* derived from the original K. The resulting cipher is called an *iterated cipher*. One of the most popular constructions for creating an iterated cipher is the *Feistel network*.

2.3.2 Feistel networks (FN)

The Feistel network was originally introduced with the Lucifer cipher, later adopted as DES, and was created by Horst Feistel while working for IBM.

The Feistel network gained popularity fast, due to the easy implementation in hardware; when decrypting, it is only necessary to invert the order of the roundkeys reducing the number of needed functions from two to one. Both block ciphers used in UMTS (Kasumi, AES) are built upon the Feistel network.

2.3.2.1 Number of rounds

The number of rounds in a Feistel networks is determined by a compromise between the security and the speed of the cipher. We want sufficient security, but the cipher cannot be too slow, because no one would use it.

2.3.2.2 Feistel network description

The FN splits the input into a left and right side (L_0, R_0), then sends the right part through the round function f, with roundkey rk_i and xors the output with L_0. Then the $f_{rk_0}(R_0) \oplus L_0$ is named R_1 and the unchanged R_0 is named L_1. This procedure is repeated until all rounds are done.

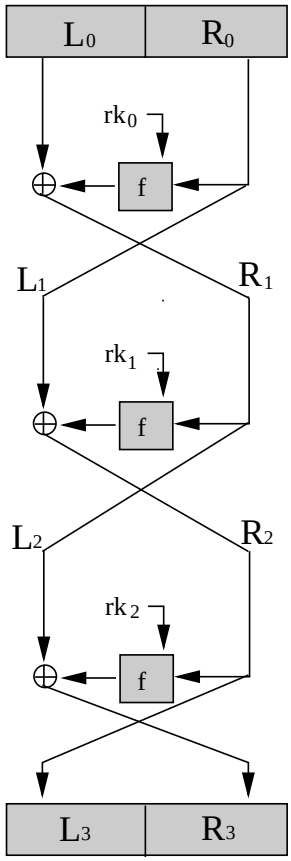

Figure 2.1: Three round Feistel network

2.4 Summary

This chapter has given a overview of what block ciphers are and an overview of the most commonly used ways for creating them. We have also seen some basic requirements of a block cipher, and seen how small permutations and substitutions can be fitted together in order to create larger, more secure ciphers using the Feistel network.

2.4.1 Conclusion

These apparently simple principles of confusion, diffusion and Feistel networks has given cryptographic designers a good "recipe" for creating secure block ciphers. Although these ciphers are not unconditional secure, many are conditional secure which suffices in most cases.

Other types of block cipher designs do also exist, but I have chosen to

emphasize this method, due to the fact that both block ciphers of 3GPP are designed this way.

Chapter 3

Pseudo-random functions and permutations

Pseudo-randomness of a block cipher is a step towards the ultimate goal of the *one notion of security*. Here we take the block cipher to a higher level by disregarding the individually possible attacks a cipher can be exposed to, and search of *a* property that can protect towards all known and unknown attacks.

This section will therefore concern about the background theory of functions, permutations and their randomness, and how they can be used to prove the security of block ciphers on a general higher level, without examining the internal of the block cipher, just the behavior.

As mentioned in section 2.3, the inspiration comes from the properties of the unconditional secure one-time-pad cipher and from Shannon's theory[4], where no information about the plaintext can be derived from the ciphertext. While the one-time-pad needs a key with the same length as the plaintext and a new key for every encryption, the output from the one-time-pad appears totally random. So when searching for the one notion of security it is preferable to keep this property, while using a smaller key. The whole idea behind pseudo-randomness is that if no one can distinguish your cipher from a random function, it is highly unlikely that any information about the plaintext leaks from the ciphertext.

In order to understand what pseudo-random functions and permutations are, and why they are so important, we need to take a look at some of the key definitions concerning the subject. My presentation is mainly based on Chapter 3 of *Introduction to modern cryptology* [2].

3.1 Function family

A *function family* is a set of functions F taking two inputs $K \in Keys(F)$ and $X \in Domain(F)$, of given lengths, and outputs $F(K, X) \in Range(F)$. $Domain(F)$ is the set of all possible input to F and is also denoted $\{0,1\}^i$, where i is the length of the input. Further $Keys(F) = \{0,1\}^k$ is the set of all possible keys to F, where k is key length. $Range(F) = \{0,1\}^j$ is the set of all possible output from the function family F with the length j.

Before deciding on a key K, F is a set of mappings from $Domain(F)$ to $Range(F)$, but when drawing a key K we create an instance of a map, denoted F_K. The formal definition is that the function family is a map $F : Keys(F) \times Domain(F) \rightarrow Range(F)$.

3.1.1 Permutation

In the case that the lengths of *Domain* and *Range* are equal, i.e the same set, and F_K is a one-to-one correspondence between the *Range* and the *Domain* sets, we call the map (F_K) a permutation. Subsequently we call $F : Keys(F) \times Domain(F) \rightarrow Domain(F)$ a family of permutations.

Example: Kasumi is a family of permutations Kasumi is a family of permutations where $Range(Kasumi) = Domain(Kasumi) = \{0,1\}^{64}$ and $Keys(Kasumi) = \{0,1\}^{128}$

3.1.2 Important families

When discussing block ciphers and random functions/permutations, there are two families that are especially important. Why these are so important is discussed in section 3.2, 3.3 and 3.4. We will define them here and they will be used throughout this chapter.

The first is the set of all possible functions, F, from the set of Domain (plaintexts) to the set of Range (ciphertexts).

$$F : Domain(X) \rightarrow Range(X) \tag{3.1}$$

The second is a much smaller family, but equally as important. It is the set of all possible permutations on the Domain set

$$P : Domain(X) \rightarrow Domain(X) \tag{3.2}$$

3.2 Random functions

A random function is a function that is randomly chosen, with uniform probability, from all possible functions in the family of functions F (See equation 3.1). And can be thought of as drawing (by flipping a coin) a random output (*Range*) for every possible input (*Domain*). i.e. for any of the values in the set *Domain* randomly select i output bits and tie together input and output.

It is important to clarify that it is *not* the output from a chosen function that is random, but it is the function that is *chosen at* random from all possible functions of the family of functions F of equation 3.1.

3.3 Pseudo-random functions (PRF)

A *pseudo-random function family* g is a function family which input/output behavior is difficult to distinguish from a random function f. By difficult we define that with polynomially many computations no one can tell which function is which (g_k or f) using the adaptive distinguisher model.

3.3.1 Adaptive distinguisher model

There are two main participants in the *adaptive distinguisher model*. We have the **Distinguisher** which task is to distinguish whether the function π is implemented as a random function f (*world* 0) or a function from the given function family g (*world* 1). The distinguisher can probe, polynomial many times, the **Oracle** with different input strings x_i from the *Domain* domain, where upon the oracle returns a string $y = \pi(x_i)$ from the *Range* domain.

The function π is chosen randomly, by the oracle, from either the f or the g function. If the oracle has chosen *world* 0 a random function from *Domain* to *Range* are drawn. If *world* 1 is chosen a function g_k is drawn by randomly selecting, with uniform probability, a key k from $Keys(g)$. The distinguisher has only access to the functions through the oracle via input and output. If the advantage the distinguisher has, for distinguishing whether y is output from f or g_k, is negligible, the block cipher is a pseudo-random function [5, 2].

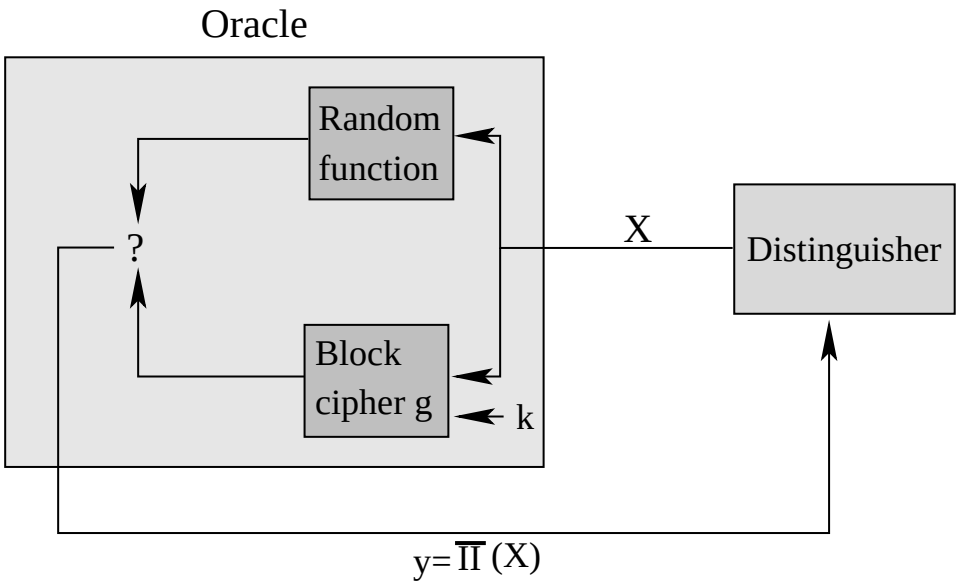

Figure 3.1: Oracle and distinguisher model

3.3.1.1 Advantage

The advantage (ADV_D) is measured by calculating the probability that the oracle (O) is in *world* 1 and the distinguisher D thinks that it is in *world* 1 minus the probability that the oracle is in *world* 0 and the distinguisher still thinks that it is in *world* 1. The advantage of the distinguisher has, is formalized in Definition 1. This definition is retrieved from reference [5] Definition 2.

Definition 1. $ADV_D = |Pr(D\ outputs\ 1|O \leftarrow f) - Pr(D\ outputs\ 1|O \leftarrow g)|$

where $O \leftarrow g$ indicates that the oracle O implements g, and vis versa.

Different distinguishers may have different advantages, but it is always the best possible distinguisher that is taken into accord when deciding whether a function is secure as a PRF. A distinguisher may be better than an other if it designed in better way, asking smarter questions, or it may use more time calculating the answers from the oracle. We are interested in the distinguishers that give the greatest advantage using the least amount of resources.

In order to find the most clever/best distinguisher it is customary to restrict the resources available to the distinguisher. These resources are often time-complexity, number of oracle queries and the sum of the lengths of the queries. By doing so we create equal terms for the distinguishers allowing us to find the best distinguisher with these limited resources.

Secure PRFs To define what advantage that is small enough for a PRF to bee deemed secure the term *negligible* is introduced. So by a *secure PRF* we mean that the advantage is *negligible* for any adversary whose computational powers are polynomially-bounded [2].

Definition 2. *To clarity what we mean with negligible I will use [5]'s version:*

A function $h : \mathbb{N} \rightarrow \mathbb{R}$ *is called negligible if for any constant* $c > 0$ *and all sufficiently large* $n \in \mathbb{N}, h(n) < \frac{1}{n^c}$.

3.4 Pseudo-random permutation (PRP)

Using the definition for a permutation it is easy to see that a family of pseudo-random permutations is a family of PRFs where *Domain* and *Range* are the same set. $F : \{0,1\}^k \times \{0,1\}^i \rightarrow \{0,1\}^i$.

Or that a family of PRPs is a family of permutations difficult (under the Adaptive distinguisher model) to distinguish from the family defined in equation 3.2.

By using our acquired knowledge about PRP and PRF we will be able to tell something about the "strongness" of a block cipher.

3.5 Block ciphers and pseudo-randomness

Every block cipher is a family of permutations F. The cipher maps a block with length i to a new block with length i, using a key K. The currently chosen key draws a function f from F ($f \in F$).

When you design a new block cipher, it is not enough for your block cipher to be a family of permutations, you also want the cipher to be a family of *pseudo-random permutations (PRP)*. The reason for this is mentioned above, and the benefits are derived from the Shared-random-function model.

3.6 Shared-random-function model (SRF)

One of the main reasons to why your block cipher needs to be a family of PRPs, is that when others are designing encryption schemes or protocols using block ciphers, they (usually) design and prove it secure in the shared-random-function model [2].

In the usual symmetric-key encryption model the participants share a short secret key, but in the SRF model the parties share a long string created from a random function, and thus it is possible to create an unconditional secure scheme (see section 2.2.1). However, a random function is too large to be stored on any computer[2], and thus the SRF becomes a *conceptual model* which models a symmetric-key cryptography system (see section 1.2) where each participant shares a random function f, which maps l bits to L bits.

In [2] they use an example with the CTR encryption scheme (See 4.3 for more details) where the E_k is substituted with the random function f. Immediately the CTR scheme is transferred to a one-time-pad with perfect secrecy. This is the essence of the SRF; create a scheme that gives perfect secrecy by using a random function.

Regrettably this f function is nonexistent so the perfect secrecy is not automatically transferable to the real world. Nevertheless, by exchanging f with your new PRP block cipher the scheme is still secure, given that the adversary does not have enough resources to do a exhaustive key search.

3.7 Summary

This section has given us the background theory in order to verity whether a block cipher is a "pseudo-random permutation" or not. If no one can separate the block cipher from a random permutation, it can be labeled *pseudo-random* and then be used with any scheme or protocol proven to be secure under the shared-random-function model. With the notion of pseudo-randomness, the scientists have taken a great step towards the ultimate goal for the modern cryptography, where we will have a *ONE* notion for the security of a block cipher.

Chapter 4

Operation Modes

Up until now, in our discussion of block ciphers, we have only discussed encryption of one plaintext block, but when the plaintext exceeds the block size, for a given block cipher, the need for operation modes emerges. If the plaintext is 140 bit long and the block size is 128, the remaining bits after the first block is encrypted are 12 bits. What are to become of the last 12 bits? Should there be a padding added to the remaining bits in order to create a whole block? Is there to be a relation between the 1^{st} and the 2^{nd} block, i.e. should the outcome of one block depend on the previous? Different operation modes have different answers/solutions depending on the intended area of usage and the designers assessments.

In this chapter I have chosen to present the most common operations modes and especially those which are used as foundations for the operation modes in UMTS. The confidentiality mode of UMTS is called $f8$, while the integrity mode is called $f9$.

4.1 Electronic Code Book (ECB)

ECB is the simplest (most straightforward) of all operation modes. Each plaintext block corresponds to a ciphertext block, similar to the old hand-written code books where one instance of, for example, a letter was replaced by another letter. The only difference is that with ECB we have a large set of "letters" to choose from.

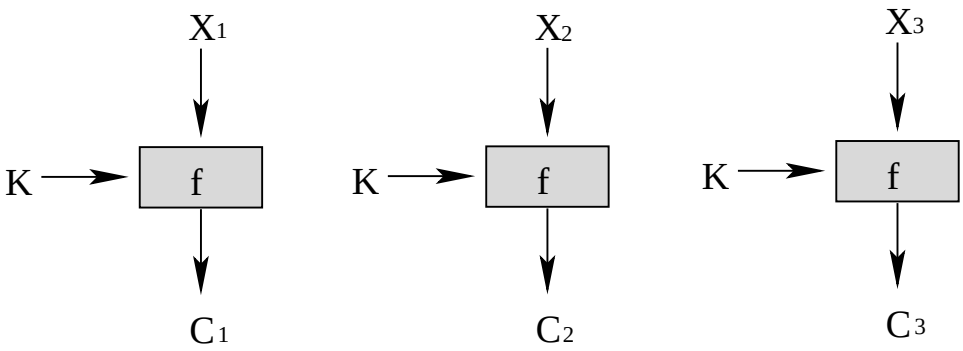

Figure 4.1: Electronic Code Book (ECB)

4.1.1 Input

The inputs to the ECB mode are the plaintext blocks x_1, x_2, ..., x_n and the secret key K.

4.1.2 Functional description

Encryption The plaintext is divided up into blocks, of size matching the block size of the algorithm used, and each block is then encrypted, with the block cipher f, individually and independently. See figure: 4.1.

Decryption Each ciphertext block is decrypted with the key K and the plaintext is found.

4.1.3 Properties

The ECB design has some advantages; Since the blocks are independent, errors are contained within the one block, and the other blocks are not affected. The design of ECB makes it possible to encrypt/decrypt multiple blocks simultaneously.

A great disadvantage is that two identical plaintext blocks will encrypt to the same ciphertext under the same key[1].

4.2 Cipher Block Chaining (CBC)

As the name indicates, this operation mode chains each neighboring blocks together, creating a dependency between a cipher block c_i and all it's pre-

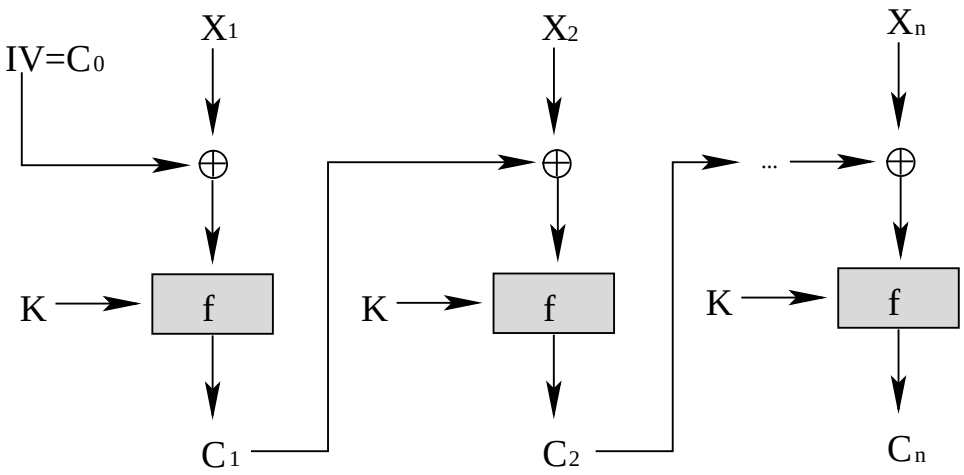

Figure 4.2: Cipher Block Chaining - Encryption

decessors. However, the first block x_1 has no predecessor, which leads to the introduction of an "initial value" (IV) [1]. The IV is used as c_0. The secrecy of the IV is not required for the confidentiality that the mode provides, although extra security may be gained from it. The correct IV must also be known to decrypt the ciphertext.

4.2.1 Input

The inputs to the CBC mode is the plaintext blocks, $x_1, x_2, .., x_n$ each of length equal to the block size of the block cipher used, the IV with size of one block size, and then the secret key K. The output is the ciphertext blocks $c_1, c_2, .., c_n$.

4.2.2 Functional description

Encryption For each plaintext block x_i, xor x_i with c_{i-1} ($c_0 = IV$) then the result sent through one round with the chosen block cipher f and becomes c_i. This procedure is continued until all n blocks are encrypted.

Decryption The decryption starts with the leftmost cipher block c_1 and it is sent through one round of f. Then the result is xored with c_{i-1} and x_i is found.

4.2.3 Properties

As discussed in the ECB section 4.1 the problem with identical plaintext blocks being encrypted to identical ciphertext blocks are somewhat reduced for the CBC mode. For collision to occur, using the CBC mode, the exact same input to the mode must be used. A single bit change in the IV, first plaintext block or the key results in that the hole ciphertext will change[1]. This is often done by implementing IV as a counter.

Since the IV is xored with the first plaintext block (or xored with the first decrypted cipher block), an adversary that changes the IV may invert some bits in this block and alter the content of this single block[6].

An error occurring in one block, during transmission, does not compromise the whole ciphertext. The correct decipherment of c_i depends only on c_i and c_{i-1}. Thus an error in c_i only influences the x_i and x_{i+1}. If c_{i+1} is received correctly, x_{i+2} will be correctly deciphered.

4.3 Counter (CTR) mode

The CTR mode is an improvement of the Electronic code book mode (section 4.1), with one additional property: Every plaintext block is xored with a string, of length equal to the block. This string is increased with one for each block, and therein becomes a counter.

This extra xoring with a counter removes one disadvantage from the ECB mode, that identical plaintext blocks result in the same ciphertext blocks.

As mentioned, the $f8$ mode adopts this property in its design. Practically the $f8$ mode is an OFB mode where each feedback S_i is xored with the counter C_i and the IV and sent as input to the kernel algorithm E_K, $E_K(S_i \oplus C_i \oplus IV$.

4.3.1 Input

The inputs to the CTR mode are the plaintext blocks and the secret key. In addition, the start value of the *counter* is needed for decryption.

4.3.2 Functional description

Encryption Each block of plaintext is xored with the *counter* and is then encrypted, with block cipher f, using the secret key, resulting in the ciphertext. The *counter* is then increased by one for each encryption.

Decryption Each block of ciphertext is decrypted with the secret key, and then xored with the *counter*.

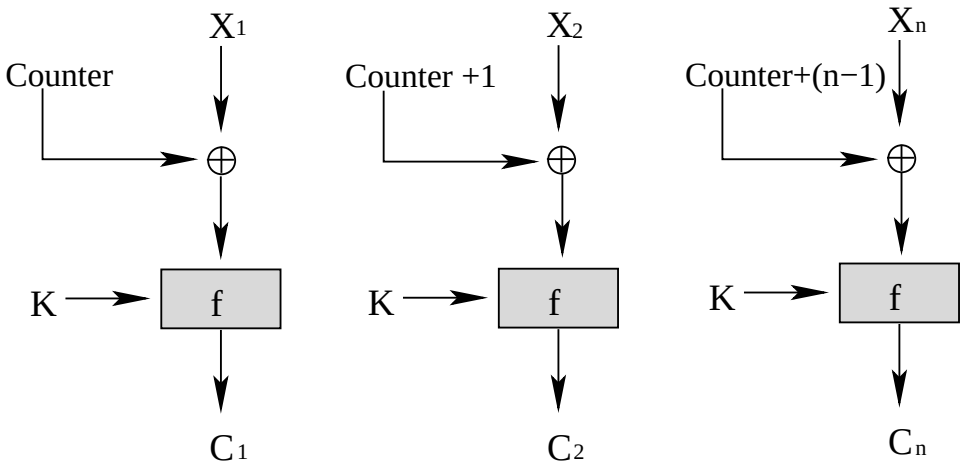

Figure 4.3: Counter mode - Encryption

4.3.3 Properties

In addition to the advantages of the ECB mode (parallel encryption/decryption, easy implementation), the CTR mode has, as mentioned, the property that identical plaintext blocks encrypt to different ciphertext blocks.

4.4 Output Feedback (OFB) mode

The Output Feedback mode represents an important branch of the operation modes; turning a block cipher into a stream cipher. This is done by specifying a number of desired bits, r, to be encrypted at a time[1].

Due to OFB properties, error correction and the preprocessing option (discussed later), the mode becomes useful when transmitting a large quantity of data over an insecure and unstable channel. Thus OFB is the preferred mode for satellite transmission.

4.4.1 Input

The inputs to the OFB mode are the secret key K, the initial vector IV and the r bit plaintext blocks (x_1, x_2, ..., x_n).

[1]usually 1 or 8 bits[1], where the 8 bits are for encrypting a whole character.

4.4.2 Functional description

Encryption Encrypt, with block cipher f, the IV using the block cipher, yielding s_1. Extract the r leftmost bits, from s_1, to be xored with the plaintext block x_1 creating the ciphertext. Send the s_1 as input to the next encryption, and continue the operation until all n blocks are encrypted.

Decryption The decryption is based on the reconstruction of the key stream (s_1, s_2, ..., s_n), and is very similar to the encryption, except that the key stream is xored with the ciphertext, not the plaintext.

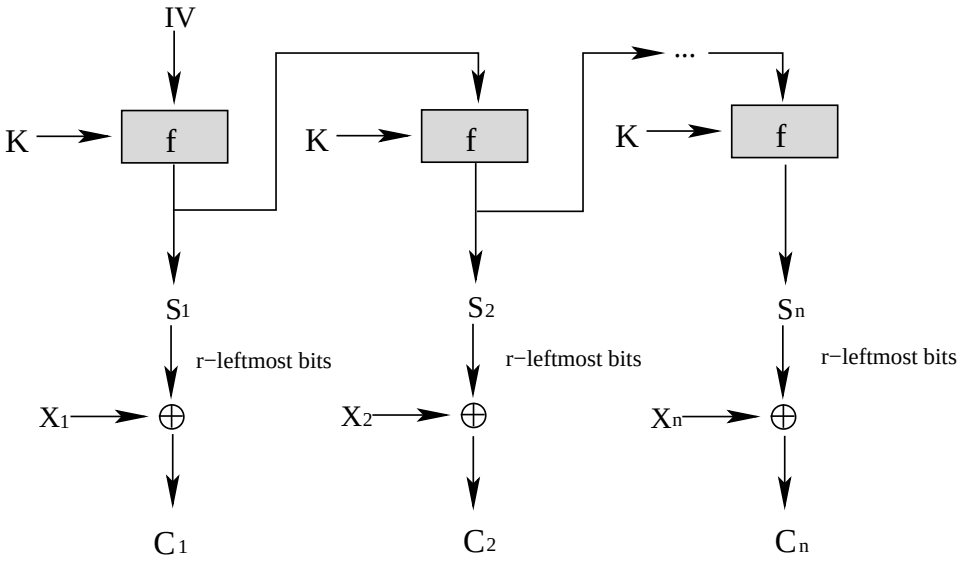

Figure 4.4: Output feedback mode - Encryption

4.4.3 Properties

The design of the OFB mode makes sure that, when transmitting data through an insecure and unreliable channel, transmitting errors or interference doesn't corrupt the whole message. This is due to that the key stream is created without knowledge of the cipher/plaintext, and an error affects just the corresponding plaintext bit(s).

It is also possible to preprocess a long string of bits (multiple of r) and then apply the xoring with the message bits, realtime. However, it is important that the IV is changed every time for the same secret key. This is due to the nature of the OFB mode where the preprocessed key stream is xored to the plaintext blocks and the same key stream are produced using the same

K and IV. For two different ciphertexts (c_i and z_i) created using the same K and IV it is possible to derive the xor difference of the corresponding plaintexts: $z_i \oplus c_i = (x1_i \oplus s_i) \oplus (x2_i \oplus s_i) = x1_i \oplus x2_i$. This is something we want to avoid.

Due to the independent calculation of the key stream, it is vital that no public key algorithm is used. If so, the regeneration of the key stream is possible with only the public known information.

4.4.4 Creating the $f8$ mode from OFB and counter mode

The OFB mode, combined with the Counter mode, creates the basis for the integrity function $f8$ of UMTS. As with satellite transmissions, the encrypted bits from $f8$ are sent through the air to the nearest receiver (Node-B, see section 6.2.2 for more details). It is therefore vital that $f8$ inherits this property from the OFB mode.

From the Counter mode the $f8$ adopts the property where each input to the block cipher is xored with the counter, and thereby adding a extra assurance that equal plaintext blocks doesn't result in the same ciphertext.

Practically, the $f8$ mode is an OFB mode where each feedback is xored with the counter and the IV. See section 6.3.3.1 and 8.4.1 for more information on $f8$.

4.5 Cipher block chaining - message authentication code (CBC-MAC)

The CBC-MAC scheme creates a "fingerprint", a MAC, of the plaintext using a block cipher. CBC-MAC is the currently most popular scheme for this purpose, and is considered secure if the block cipher is a pseudo-random permutation[7].

The resemblance to the CBC mode is apparent (see figure 4.5), the only difference is that a MAC is created by extracting just the last encrypted block.

The input to the CBC-MAC mode is a string of any length. While the output is equal to one block length of the used cipher f.

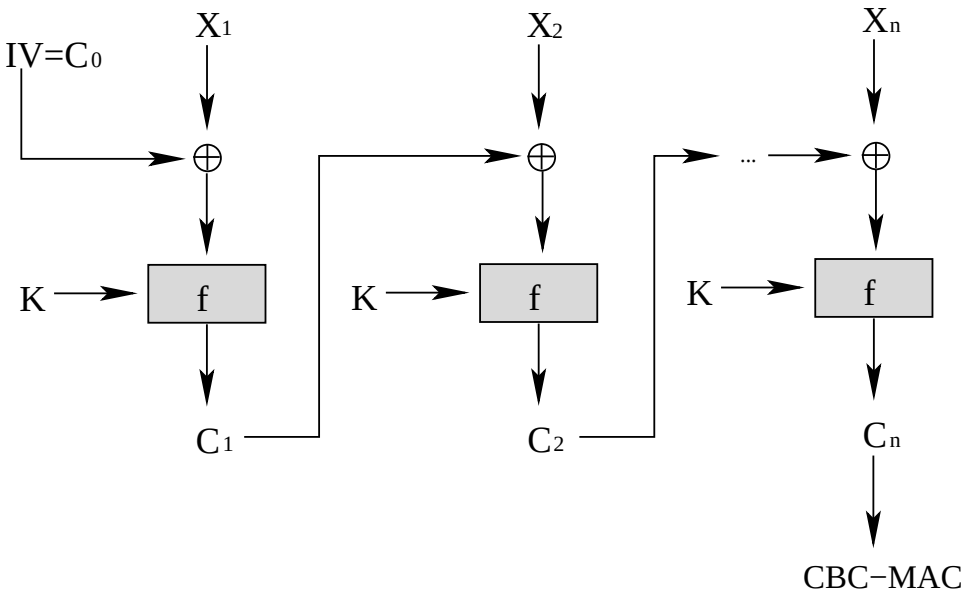

Figure 4.5: Cipher block chaining - Message Authentication Code

4.5.1 Creating the $f9$ from the CBC-MAC mode

The design of $f9$, the integrity mode of UMTS, is based on the CBC-MAC. $f9$ has some additional computations (xor) where each $c_1 \rightarrow c_n$ (see figure 4.5) is xored together and sent through one extra round of the block cipher. This is done to add some extra security to the $f9$ mode. See section 6.3.3.2 and 8.4.2 for more information about the $f9$ mode.

4.6 Summary

This chapter has given an overview of some available operation modes when providing both integrity and authentication using block ciphers. Other modes do also exist, but the selected modes in this chapter are used as foundations for the modes of UMTS, and thus given a priority.

Some modes create a key stream that is xored with the plaintext, other send the plaintext through the cipher. One must choose a mode that suites the intended area of use, not all modes are suitable for all purposes. An example is the OFB mode where it has a great advantage when the information is transmitted over unstable channels, as the air, due to the fast error recovery.

Operation modes are very important tools when providing encryption and authentication, and it is important to be aware of that an excellent cipher

is nothing without a proper operation mode.

Chapter 5

Introduction to cryptanalysis

Cryptanalysis is the second main half of the term *Cryptology*. It is vital for the user of an integrity system to know that no method of analysis exists able to break the cipher. Therefore there is an ongoing struggle between the crypt designers and the crypt analysts to come up with an immune cipher or a new powerful attack (respectively).

To clarify the meaning of the notion *cryptanalysis*, I would like to present the definition retrieved from [1] page 15:

> Cryptanalysis is the study of mathematical techniques for attempting to defeat cryptographic techniques, and, more generally, information security services.

In other words, cryptanalysis tries to undo the work done by the cryptographic cipher either by finding the plaintext (the original text) or the key which was used. Given the key an adversary can always find the plaintext.

Due to the wide variety of different cryptographic techniques there also exists a wide variety of different counter techniques. However it is a usual assumption that the opponent knows which cryptographic method that is used. This is usually called *Kerchoff's principle* or *Kerchoff's assumption*[1]. Without this knowledge the job of a cryptanalyst becomes harder. It is also accustomed for the opponent to have access to all the data sent between the participants in the symmetric-key encryption scheme (see section 1.2.

This chapter will first discuss the classes of attacks that exist, where each class gives restrictions to what kind of information that is available to the adversary. Further, the chapter will give a general overview of different attacks that are commonly used to analyze block ciphers, and the properties (of block ciphers) that the attack exploits.

5.1 Classification of attacks

There are to main classes of attacks, active and passive. Where in the *passive attack* model the adversary only observes the information flow and tries to retrieve knowledge about the observed information offline. While in the *active attack* model the adversary actively tries to change, delete or add information to the flow[1].

5.1.1 Passive attacks

The passive attacks are split into four classes. The goal of these attacks is to find the key, primarily, secondly to find the plaintext. The first attack (*ciphertext only*) is the hardest, where the analyst gains little information, but for each new attack he gains more and more information. This escalates to the *chosen plaintext attack*, where the adversary can probe the cryptographic cipher with his own texts. The four classes are:

ciphertext only The adversary knows only the ciphertext and which cipher is used for encryption.

known plaintext In addition to the *ciphertext only* attack the opponent now also knows the plaintext, or parts of the plaintext, that where encrypted. This attack is not unlikely, due to the fact that we may often know some initial standard coding of the plaintext. One example is a html file which often starts with "$< html >< head >< title >...$" etc.

chosen plaintext Here the adversary can also, in addition to the properties of a known plaintext attack, get his plaintexts encrypted under the current key.

chosen ciphertext In the final class of attack the opponent can choose ciphertexts that he/she wants decrypted with the correct key.

A cipher secure from a given attack is also secure against those attacks using less information. I.e. a cipher secure against a chosen plaintext attack is also secure against known plaintext attacks and ciphertext only attacks.

5.1.2 Active attacks

The class of active attacks is a less used class compared to the passive, when attacking a block cipher. This is due to the fact that the opponent is actively changing the information sent through the channel, and is therefore more easily compromised. The active attacks are nevertheless used when

attacking encryption protocols or schemes, but not against the block cipher alone.

As the attacks studied against Kasumi are all passive attacks, the topic of active attacks will not be discussed further outside this section. However, in section 8.4.4.2 a forgery attack against $f9$ is discussed.

5.2 Types of Passive attacks

5.2.1 Definitions

Before discussing the details concerning each type of attack, some basic definitions needs to be clarified to better understand differential cryptanalysis. First is the notion of a characteristic and then a differential, both which are properties of an iterated block cipher. Differential attacks depend on these properties to exist.

5.2.1.1 Characteristics

Take any two plaintexts and xor them together, the result will be a **xor difference**. *A one-round-characteristic* is created from a pair of plaintexts (P_1, P_2) and the corresponding pair of ciphertexts (C_1, C_2), i.e. input and output of the function E. Then take the xors of the plaintexts (Δ_P) and the xor of the ciphertexts (Δ_C) and you will have what is called a one-round-characteristic. I.e. a one-round-characteristic is the tuple (Δ_P, Δ_C) [1].

$$\Delta_P = P_1 \oplus P_2 \tag{5.1}$$

$$\Delta_C = C_1 \oplus C_2 \tag{5.2}$$

A *n-round-characteristic*, in addition to the one-round, also includes the xor differences (Δ_{IM}) of the intermediate rounds in a Feistel network. The Δ_{IM} is in itself a tuple. If the Feistel network has i rounds, then Δ_{IM} has $i-1$ items. The n-round-characteristic is the tuple (Δ_P, Δ_{IM}, Δ_C). See figure 5.1.

The probability of a characteristic is the probability that a randomly chosen input pair with a fixed xor difference, produces the "correct" ciphertext- and intermediate round differences. [8].

[1] In practice one selects just a random P_1 and then xor P_1 with the desired Δ_P to obtain the P_2

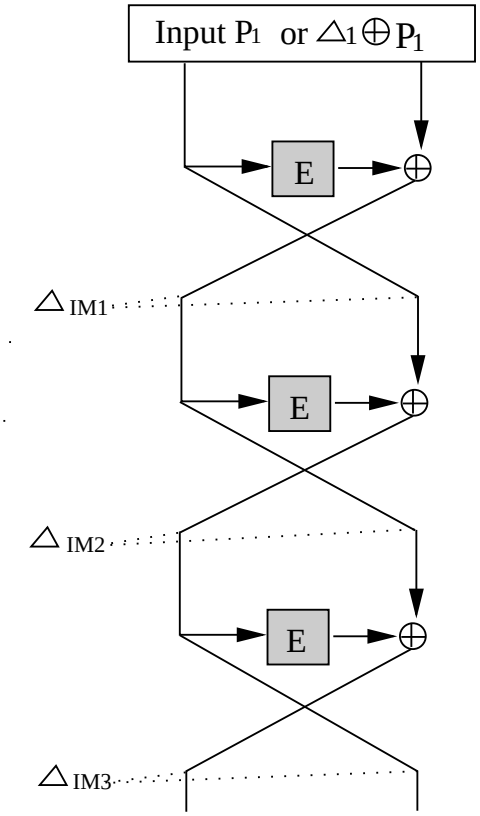

Figure 5.1: N-round-characteristic

5.2.1.2 Differentials

A differential is more or less a characteristic, or a part of a characteristic, but a differential disregards if the characteristics of the intermediate rounds are followed or not. By definition in [9] a n-round-differential (i.e. differential) is the tuple of (Δ_P, Δ_N), where Δ_N is the xor difference of the output of the n-th round.

The probability of a n-round-differential is the probability that a randomly chosen input pair with a fixed xor difference, produces the "correct" output difference after the n-th round.

5.2.2 Differential cryptanalysis

Differential cryptanalysis is a chosen plaintext attack against an iterated block cipher f, introduced in 1990 by Biham and Shamir[10]. It requires lots of input and output pairs, where each pair has a given xor difference. The pairs are often collected in tuples (X, X', Y, Y'), where X and X' are inputs

with the given xor difference ΔX, encrypted under the same, unknown key K, deriving Y and Y', respectively.

After collecting the large number of tuples we decrypt the last round of f with a possible subkey for all the tuples for both X and X' deriving C and C' respectively. If the decryption yields the right *differential*, $C \oplus C'$ then this subkey's count is increased by one. After running through the probable subkeys, the subkey with the highest frequency is probably the correct subkey.

Differential cryptanalysis depends on the probability of a n-round-differential being significant, i.e. large enough so that the amount of needed tuples does not grow to large. If the complexity (time consuming data collection and calculations) of a differential attack outgrows the complexity of an exhaustive key search, no one will attempt a plain differential attack.

5.2.2.1 Differential cryptanalysis using impossible differentials

This type of analysis uses differentials with zero probability, to find subkeys that are *incorrect*. First select a huge amount of plaintexts with a difference that is impossible to occur. Select a subkey, and check the i'th round (usually $i = n - 1$, where n is the number of rounds in the cipher) of the cipher if an impossible difference has actually occurred. If so, you will know that your subkey was incorrect and you can remove it from your list of possible subkeys [11]. It is then possible to either reduce the list to only one remaining subkey, or discard the majority of the subkeys and perform an exhaustive search on the remaining subkeys.

5.2.2.2 Xor-restricted related-key attacks(RKA)

The *xor-restricted related-key attack* is a chosen plaintext attack where you are allowed to get your plaintext encrypted with a pair of related keys, K and K'. These two keys differs normally only in one bit or.

5.2.2.3 Higher order differential attacks

A standard differential attack has order 1, $E_k(x) = E_k(y) \oplus W$ where $y = x \oplus a$, k is the key to the function E. A differential of order 2 is where $E_k(x) \oplus E_k(y) = E_k(z) \oplus E_k(v) \oplus W'$, where, in addition, $z = y \oplus a'$, $v = z \oplus a''$. Where a, W, a', a'' are any strings.

So by using differentials with higher order (than 1) it is possible to find other differentials with higher probability.

5.2.3 Linear cryptanalysis

Linear cryptanalysis was first introduced by Matsui in 1993 [12], for the DES cipher as a known plaintext attack. Linear cryptanalysis uses linear approximations between a subset of plaintext bits and a subset of the output bits of the final substitution (before the last key mixing). The linear approximation is calculated for the xors of some subset bits of the plaintext, which gives a probability for one output bit, having the value 1, away from $\frac{1}{2}$.

The course of action is as follows: Find the linear approximations, guess on a subkey, do a reverse keymixing with this subkey with your ciphertexts. Then check and see if your linear approximation holds. If so increase the count of this subkey with one. After running through all our plaintext/ciphertext pairs, the subkey with the highest count is probably the correct one.

Linear cryptanalysis depends on the possibility of the creation of a linear approximation between some input bits and some output bits. Without a high probability for the creation of this approximation, linear cryptanalysis becomes to complex and demands too many computations.

5.2.4 Side channel attacks

Side channel attacks exploit information that the hardware crypt-unit gives up, other than the usual inputs (plaintext) and outputs (ciphertext). A hardware crypt-unit uses electricity and time, and radiates heat. By manipulating additional input, such as the voltage to the unit, or simply by monitoring the time the unit uses for each operation, one may be able to withdraw information from the unit which was not originally intended by the designers. Side channel attacks can also be used in addition to other "regular" attacks.

5.2.4.1 Timing attacks

A timing attack measures the time the unit uses for its operations, and uses this information as input to a statistical model that may be able to guess some key bits. Type of hardware (RAM, processor) and other variables must be taken into consideration for the model to be as accurate as possible.

5.2.4.2 Simple power analysis (SPA)

Power consumptions attacks exploit the fact that all hardware units needs electricity, and produce electromagnetic radiation. The attack, when used

in accordance with other regular cryptanalysis, may recover the secret key.

Simple power analysis uses a device (resistor) which is able to monitor the units power consumption while performing an encryption. By interpreting this information the adversary may be able to identify what kind of instructions that are used, the order that they are performed in etc.

5.2.4.3 Differential power analysis (DPA)

Differential power analysis is similar to the Simple power analysis, but uses a more sophisticated approach, and may therefore be harder to prevent. In addition to the visual attack of the SPA, the DPA also uses statistical analysis and methods in order to gain extra information about the current secret key.

5.3 Summary

This chapter has tried to give an introduction of the most commonly used attacks of today. Especially the differential and linear attack are important and widely used. They are also probably the most effective methods of analysis that exists.

Chapter 6

UMTS - Universal Mobile Telecommunications System

6.1 Introduction

In the evolution of mobile technology we are now entering the third generation, known as the 3G. For the earlier generations (1G and 2G) the number of different systems and standards were many. Different regions had different mobile systems and technology, and travelling outside your region with your cell phone could easily prove difficult. Though we saw a big improvement from the first to the second generation, the situation was far from perfect. (i.e. one global system)

In 1998, a group of telecommunications standard bodies joined in a project which goal was to develop specifications for a global mobile system. This project was named Third Generation Partnership Project (3GPP). The specifications for 3G were to be based on the specifications and reports from the existing 2G Global System for Mobile communication (GSM) core network, and the Universal Terrestrial Radio Access (UTRA)[13]. The sum of all these specification has, in Europe, been given the name Universal Mobile Telecommunications System (UMTS) by the European Telecommunications Standards Institute (ETSI). In USA and Japan it has been named International Mobile Technology (IMT-2000), by the International Mobile Union.

This chapter will try to give an overview of UMTS architecture, security architecture and in addition give the reader an understanding of *what* we need to protect (security features), where we want that protection and how the protection is implemented (security mechanisms).

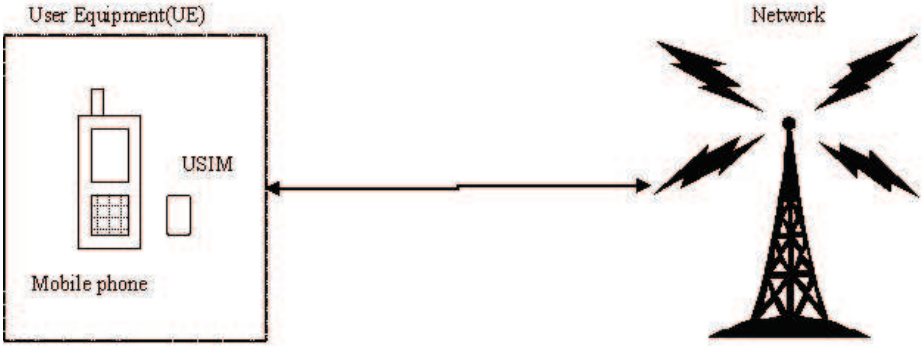

Figure 6.1: The UMTS architecture from the users point of view

6.2 UMTS architecture

6.2.1 Designing the UMTS architecture

When designing the 3G/UMTS architecture the designers wanted to reuse, and remain compatible with, the 2G/GSM architecture. Using the 2G as a building block the designers kept the features proven to be necessary and secure enough for 3G, while adding new and modifying others, in order to meet the demands of the 3G and it's new network architecture[14].

6.2.2 The architecture

To an end user of UMTS the architecture seems quite plain and simple. See figure 6.1. You just go to your local supplier of electronic devices and purchase your User Equipment (UE). Normally the UE will consist of a UMTS mobile phone (ME) and a UMTS Secure Identity Module (USIM). Other types of UEs are also possible depending on which services and applications that will be provided. The USIM is user dependent and holds the information needed to connect to and be authorized by a UMTS network. The USIM is implemented, as an application, on a Smart card called Universal Integrated Circuit Card (UICC). UICC can hold multiple USIM's, together with their encryption and authorization keys, or store other relevant applications.[14]

As we explore the network things gets more complicated. What we up until now has called "the network" consists of two underlying domains. The first domain we encounter is the UMTS Terrestrial Radio Access Network (UTRAN or AN). Its task is to physically connect the USIM, in the UE domain, with the UMTS service provider. UTRAN makes sure that the signals,

sent through the air, is converted and delivered to your UMTS provider via the antenna/base station (Node-B/BS) and the Radio Network Controller (RNC). The RNC controls several Node-Bs and serves as an access point for the Core Network (CN) domain.

The CN does also consist of multiple smaller parts. One of the first tasks it performs when a UE tries to log on, is to authenticate the USIM. The information needed in order to perform the authentication lies in the Authentication Center (AuC), which is seated in the Registers Home Environment (HE). In fact the AuC does also hold the information needed to conduct several other security services such as confidentiality, decryption and integrity. See section 6.3 for more details. AuC is the opponent of the USIM, and must contain enough information in order to perform the mentioned tasks.

The HE does also contain a list over UEs that are not welcome, i.e. the USIM are not allowed to be authenticated to the AuC because the ME is black listed. This list is called the Equipment Identity Register (EIR). The third part/register of HE is the Home Location Register (HLR). The HLR stores user profile information and the users current location (which Node-B the UE is currently connected to).

Besides the HE, CN does also consist of the Circuit Switched (CS - voice) and Packet Switched (PS - data) domain. These two domains uses the HE when providing and supporting a wide variety of services to the end user. The CS and PS domain are intended to be as universal as possible in order to handle the variety of services and changes (additions) that may surface in the future.

The last functionalities of the HE, that we discuss here, are those provided by the Serving GPRS Support Node (SGSN), Mobile Services Switching Centre (MSC) and the Visited Location Register (VLR). SGSN is seated in the PS domain and performs the services when the USIM is in PS mode, and visa versa the MSC in the CS domain. SGSN/MSC serves the 3G and 2G connections respectively, and they use the VLR when performing the authentication and key agreement, between the RNC and the USIM (see section 6.3.2.1 for further details).

6.3 Security Architecture

The section 6.2 gives a short introduction to the UMTS architecture, but it hardly mentions the security aspects. While 6.2 talks about the physical components, the security architecture lies in the communication between the components, and can be thought of as the sum of all security features and mechanisms “protecting” the communication [15]. When protecting

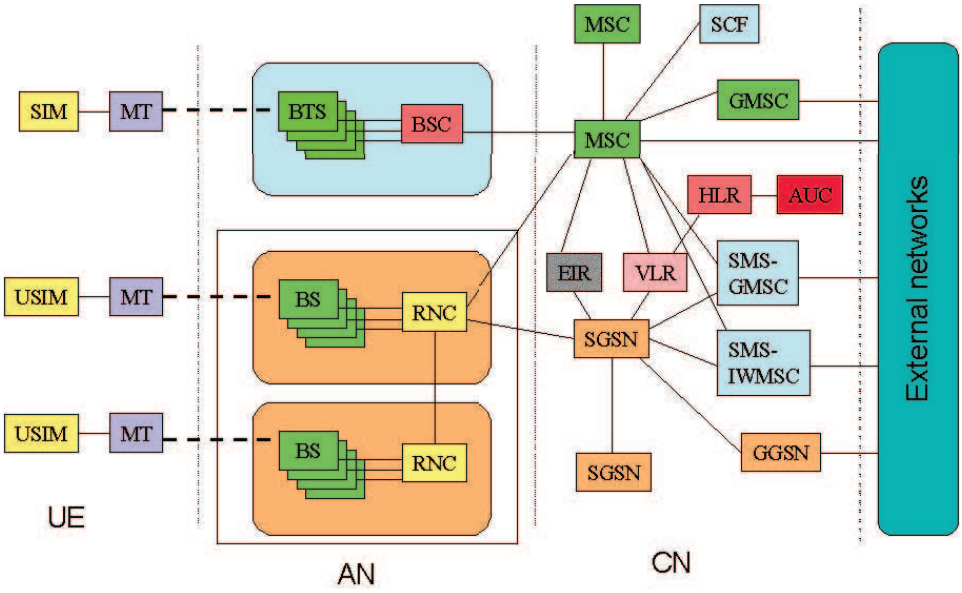

Figure 6.2: The UMTS architecture

information three keywords keep reappearing; confidentiality, integrity and authentication. Here the UMTS security does not differ. The difference, however, lies in *where* and *how* these terms are applied. The security features of the UMTS security architecture deals with where we need to apply security, and the security mechanisms tells us how we implement these desired features. This section will give an overview of the security features and mechanisms, and hopefully, give you a better understanding of what the designers of 3G wanted to protect/secure and the means they used to accomplish their goals.

6.3.1 Security Features

The 3GPP has conducted an assessment on the security threats to the 3G system and came up with a list of the most significant types of threats (see [16] for more information). Based on this assessment they have defined what kind of protection we need where in order to withstand possible attacks and meet the security requirements.

6.3.1.1 User domain security

One of the threats that was rated most significant were the use of a stolen USIM/UICC/ME, i.e. unauthorized users gaining access to another users

account and/or terminal. Thus, when designing the security features, the 3GPP added a feature that concentrated around the User domain, i.e. the security between the entities that the user has physical contact with. This resulted in that the user must authenticate to the USIM and the USIM must authenticate to the mobile terminal (ME).

6.3.1.2 Network access security

The results from the risk assessment showed that the need for several features protecting the radio interface was imminent. Hence the features in the Network access security are supposed to take care of the security between the user domain and the 3G services with emphasize on the radio access link. By security we mean protection from eavesdropping, masquerading as a network component (USIM, Node-B, RNC), passive traffic analysis etc. [16].

User identity confidentiality Each user has his own unique digital representation of his/hers identity called International Mobile Subscriber Identity (IMSI). This identity needs to be protected from masquerading, replay etc, and three features are described in [15] in order to provide the necessary security.

user identity confidentiality The first property has the task of hiding the identity from eavesdropping on a radio link when the user is accessing a service.

user location confidentiality The "user location confidentiality" is supposed to protect the user from "announcing" his/hers arrival to a specific area, through the radio link.

user untraceability And the final user identity confidentiality property shall protect from an adversary who's goal is to find out what kind of services a given identity is accessing.

Entity authentication As the headline suggests, this paragraph is focused on the authentication of the entities. The entities in this context are the USIM in the UE and the serving network. A major upgrade from 2G is that the authentication runs two ways, i.e. that the network knows the "true" identity of the user, and the user knows that the current network and it's services are authorized by the user's Home Environment (HE).

Confidentiality To protect the information sent between the UE and the RHE, we need some form of confidentiality, and the four features below

address the issues discovered in the risk assessment.

cipher algorithm agreement Two parties, the UE and the RHE, has the need for agreement, in a secure matter, on a cipher algorithm that; they both have knowledge of and one that they are going to use until their next agreement.

cipher key agreement As above, but now they need to exchange a new cipher key.

confidentiality of user data The property that it's possible to protect the user data transmitted to/from the radio access link.

confidentiality of signaling data The signaling data must not be overheard on the radio access interface.

Data Integrity The features of authentication and confidentiality are not in itself enough to provide complete security. The missing ingredient is a conformation that the data received has not been tampered with. The features in this paragraph will give an outline of the integrity we need.

integrity algorithm agreement Two parties, the mobile station (MS) (MS consists of the pair ME and USIM) and the current Serving Network (SN), has the need for agreement, in a secure matter, on an integrity algorithm that; they both have knowledge of and one that they are going to use until their next agreement.

integrity key agreement As above, but now they need to exchange a new integrity key.

data integrity and origin authentication of signaling data The property that the receiver of information knows for sure that the data has not been tampered with in any unintended way.

6.3.2 Security Mechanisms

Based on the security features, the 3GPP has defined a bundle of mechanisms that will fulfill the requirements and provide the necessary security. The foundation for all other security mechanisms is the Authentication and key agreement, from which all keys are derived.

6.3.2.1 Authentication and key agreement (AKA)

Many of the security mechanisms of 3G depend on keys distributed only between the RNC and the UE. The AKA is to provide the entities with a

secure distribution of the keys given that the UE and the RNC has achieved mutual authentication. The 3G AKA is based on a challenge/response protocol, but it gets some help from a sequence number-based one-pass protocol [15].

In order to perform the AKA the entities need some common secrets (information). Below you will find a list of the common information shared between the AuC and the USIM.

- user specific secret key
- MAC functions $f1$, $f1*$
- Response generator function $f2$
- key generation functions $f3$, $f4$, $f5$

Based on this information we are now able to perform the AKA between RNC and the USIM. Some important "by-products" results from the AKA and are generated from the key generation functions $f3$, $f4$, $f5$, respectively. These are the integrity key (IK), cipher key (CK) and the anonymity key (AK). Immediately after an AKA, the USIM and the RNC start using these new keys.

AKA Challenge/Response overview The mutual authentication of RNC and UE is triggered by the RNC. When the need for an AKA arises, the RNC sends a request to the VLR. The VLR responds with an Authentication Vector (AV) containing authentication information and the confidentiality and integrity keys. The AV is previously generated by the AuC for a given IMSI. After receiving the response from the VLR, parts of the message ($RAND||AUTN$) is sent to the UE. Based on these parts the UE is able to authenticate (or reject) the SN, and generate a response (RES) with the function $f2$. Upon receiving the answer from the UE, the RNC compares the response with the expected response (XRES), derived from the AV. If they match then the UE has also been authenticated.

By using the challenge/response protocol along with the fx functions, one achieves the AKA just (more or less) by sending a random number through the radio access link. This means that no one can produce the IK, CK or AK without knowledge of the user secret key.

In the introduction to the AKA I said that the challenge response protocol receives some help from a sequence number. By adding the SQN number to the challenge/response protocol, and storing previously used SQNs in the USIM and the VLR, the AKA gets more robust and can withstand possible replay attacks.

Generation of the authentication vectors It's the AuC that has the required information needed to create the Authentication Vector (AV) needed in the AKA. The AuC combines this information, shared between the AuC and the UE, and several functions to pile up a list of AVs by increasing the SQN number. The list is then stored in the VLR such that the AuC doesn't need to be contacted every time the UE is to be authenticated. See figure 6.3. The steps the AuC performs to create one AV are:

1. Generate a NEW sequence number (SQN_{AuC})
2. Generate a random number ($RAND$)
3. Use the $f1$ function with input SQN, AMF [1] , $RAND$ and the user specific key (k) to generate a MAC
4. Based on the $RAND$ and the functions $f3$, $f4$, $f5$ generate a confidentiality key (CK), an integrity key (IK) and an authentication key (AK), accordingly.
5. the $f2$ generates the expected response ($XRES$) from the UE.
6. Concatenate the strings $AUTN = SQN_{AuC} \oplus AK||AMF||MAC_{AuC}$ and $AV = RAND||RES||CK||IK||AUTN$

Both the USIM and the AuC keeps track of the previous used SQNs, but they allow "old" unused sequence numbers within the last 32 SQNs. This is to ensure that synchronization failure doesn't happen to often.

Authentication vectors in the UE After receiving the AV vector from the VLR, the RNC sends $RAND||AUTN$ to the UE/USIM. The USIM must perform the tasks, from previous paragraph, in a slightly different order than the AuC to retrieve the information. See also figure 6.4.

1. From the $RAND$, generate the AK with the $f5$ function.
2. Calculate the SQN_{USIM} by xor'ing the $SQN_{AuC} \oplus AK$ with the generated AK
3. Generate IK, CK, RES and MAC_{USIM} and verify that your MAC_USIM is identical with the received MAC_AuC. This implies that your key k, $RAND$ and SQN is identical to the AuC versions.
4. Verify that the SQN is in range.

[1] AMF is used to defining operator specific options during the authentication

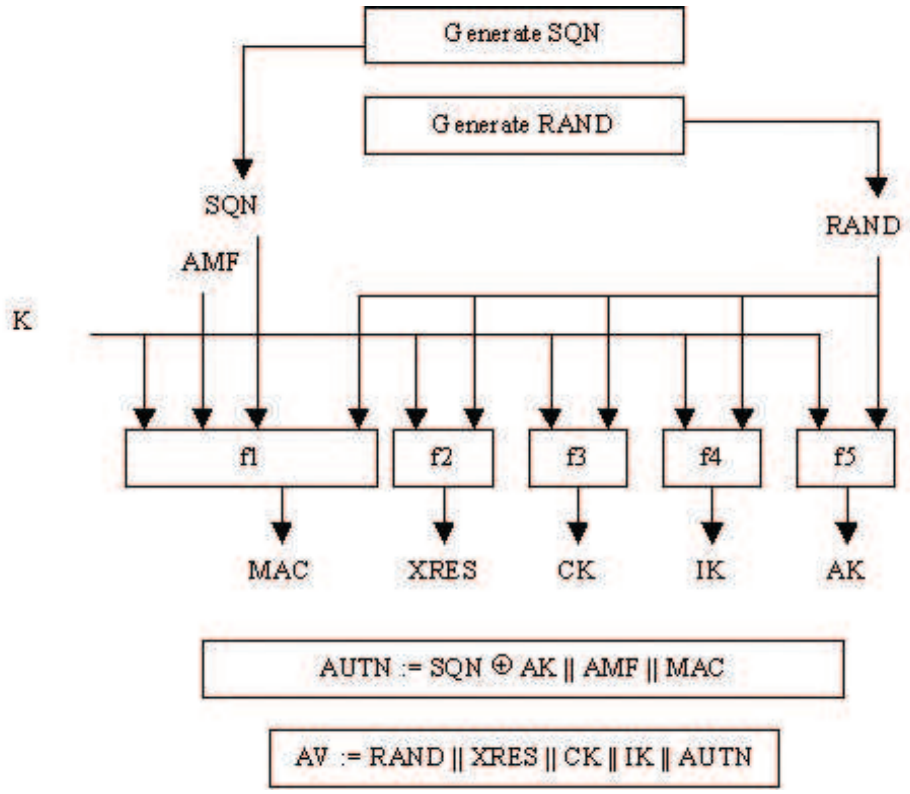

Figure 6.3: Generation of the authentication vectors [15]

6.3.2.2 The AKA functions

Previously I have only mentioned the functions that are used in the AKA of the UMTS, but in this section we will dig a bit deeper.

The functions are build as a framework and are based on a 128 bit block cipher and a 128 bit Operator Variant Algorithm Configuration Field (OP) [17]. Both can be individually specified by a UMTS provider/operator, but the 3GPP provides an example implementation, Milenage, using the block cipher named Rijndael. See chapter 9 for more details about Rijndael, and chapter 10 for the example implementation.

3GPP also provides some guidelines for choosing a secure enough block cipher. The value of OP, however, is entirely up to each provider, but it provides a separation between different providers for the fx functions. The OP value is not used directly, but a OP_C value is computed by sending the OP value through the chosen block cipher. We denote the block cipher E_k.

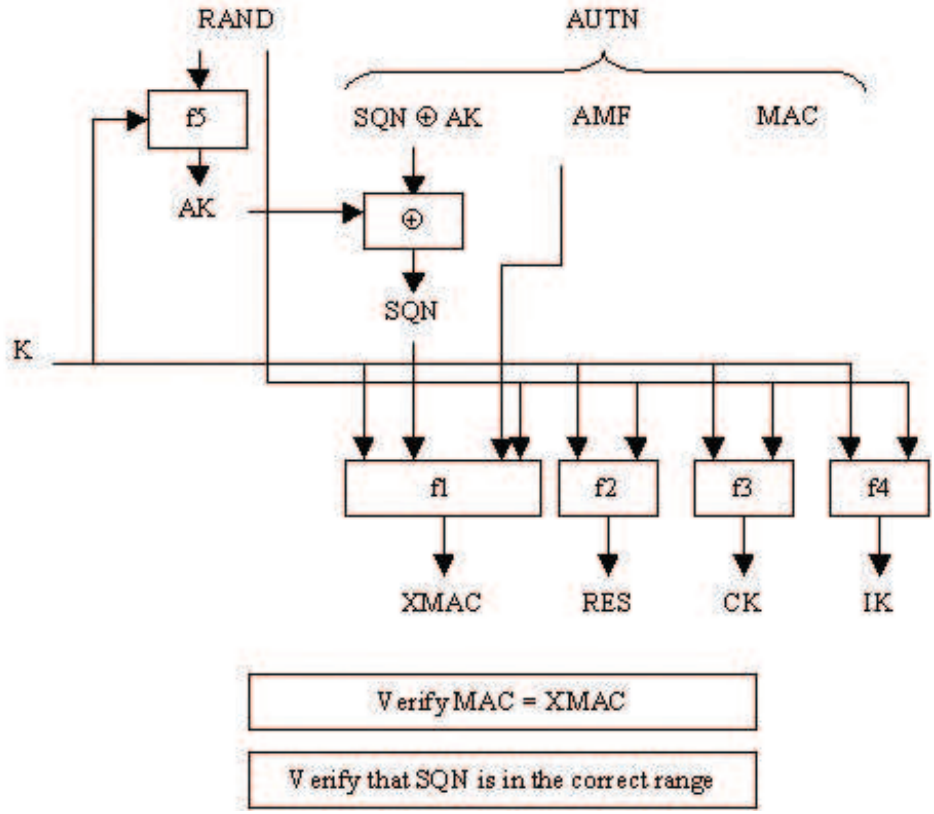

Figure 6.4: User generation of the authentication vectors [15]

General 3GPP guidelines for the AKA functions It should be computational infeasible to calculate the secret key K from the input and output of the functions. Further it should be possible to exchange the kernel function.

Design criteria

- The functions should be secure for the next 20 years. Secure means that exhaustive key search is infeasible, and that no attack, with fewer computations than exhaustive search exists. [18]
- It should be possible for each UMTS provider to personalize the example set. [19].
- The functions should be designed in such a way that the kernel cryptographic block cipher could be replaced, by any other suitable cipher [19].

Requirements for the block cipher

- Key size of 128 bit, input size of 128 bit.
- Resistance against side channel attacks
- Efficient implementation on a smart card with a 8 bits processor.
- It can be replaced with any other kernel function, matching the above criteria.

$f1$: The $f1$ function is the message authentication function (MAC). Its job is to create a "finger print" of a message mixed with the secret user key by using the block cipher function E_k. By applying a MAC,to a message, the receiver knows that only a entity with knowledge to the secret key K can create the MAC, i.e. the receiver can verify that the message is authentic and comes from the entity that claims to have sent the message.

The inputs to the function, except the key, are the 128 bit random number RAND, generated by the $f0$ function, the 48 bit sequence number (SQN), a 16 bits authentication management field (AMF). The output from the function is a 64 bits authentication code.

$f1*$: The $f1*$, is identical to the $f1$, i.e. the same input and output, but $f1*$ is only used when the SQN number is re-synchronized.

$f2$: The inputs to the $f2$ function are the 128 bits secret key K and the 128 bits random number RAND. The function mixes these two together, by using E, such that the output is a unique combination of the inputs. $f2$ is used to generate RES and XRES such that the UE and the RNC, during the AKA, can prove that they are who they claim to be.

$f3$: Based on the key K and the random number RAND the $f3$ derives a cipher key (CK) with the help from E_k. The CK is the confidentiality key and is used as input to the encryption function $f8$.

$f4$: Identical to the $f3$ function but $f4$ generates the integrity key (IK). IK is used as input to the integrity function $f9$.

$f5$: Identical to the $f3$ function but $f5$ generates the Anonymity Key (AK). AK is used when masking (xor) the SQN when it is transmitted through the radio access network.

$f5*$: Identical to the $f5$ function but $f5*$ is only used when the SQN number is re-synchronized.

6.3.2.3 Identification by temporary identities

As the three features in paragraph "User identity confidentiality" in section 6.3.1.2 states, we need to protect the IMSI. To accomplish this, the VLR creates a Temporary Mobile Subscriber Identity (TMSI) based on the IMSI. Only the VLR has access to the link between the IMSI and the TMSI. As the TMSI is only valid within the area where the user is registered, the TMSI needs to be accompanied by a Location Area Identification (LAI) outside this area. So by using the pair TMSI/LAI, we are now able to identify the user without sending the IMSI.

When it is not possible to be identified by a TMSI/LAI pair, the VLR may request the ME/USIM to identify itself by the IMSI. This happens the first time the user register in a SN or when VLR cannot retrieve the IMSI from the TMSI. This mechanism, however, does not fulfill the features in 6.3.1.2.

6.3.2.4 Initiating integrity and confidentiality

The AKA, see 6.3.2.1, produces an integrity and encryption key, IK and CK respectively. Immediately after the first AKA it is possible to perform both integrity check and encryption, given that the identity (IMSI/TMSI) of the USIM is known. But first, the two entities (MS and RNC) need to agree on a common algorithm both for the integrity and the confidentiality. The USIM provides the VLR with a list of algorithms it supports, whereupon the VLR chooses the preferred algorithms.

Each pair of IK and CK is given a Key set identifier (KSI) such that when the pair is to be reused only the KSI is exchanged.

6.3.2.5 Lifetime of CK and IK

In addition to new keys after each new AKA, the CK and IK keys are protected with a mechanism that restricts each key extended use. The threshold is set by each operator and stored in the USIM. If the encrypted information under the current CK exceeds the threshold, the CK and IK are deleted, and thus forcing the creation of a new AKA.

6.3.3 Confidentiality and integrity functions

On the contrary to the AKA functions, the confidentiality and integrity functions are specified by the 3GPP as $f8$ and $f9$ respectively. These functions use the Kasumi algorithm as a building block to provide the required services between the UE and the RNC.

One reason to why the AKA functions are not specified, while the $f8$ and $f9$ are, is that the UE connects to other operators Node-Bs and RNCs when the UE's own network is not available. It is therefore vital that the confidentiality and integrity is kept when entering other networks than your own. With the AKA, it is always your own Home Environment that is contacted and that performs the AKA, while the confidentiality and integrity is terminated at the RNC.

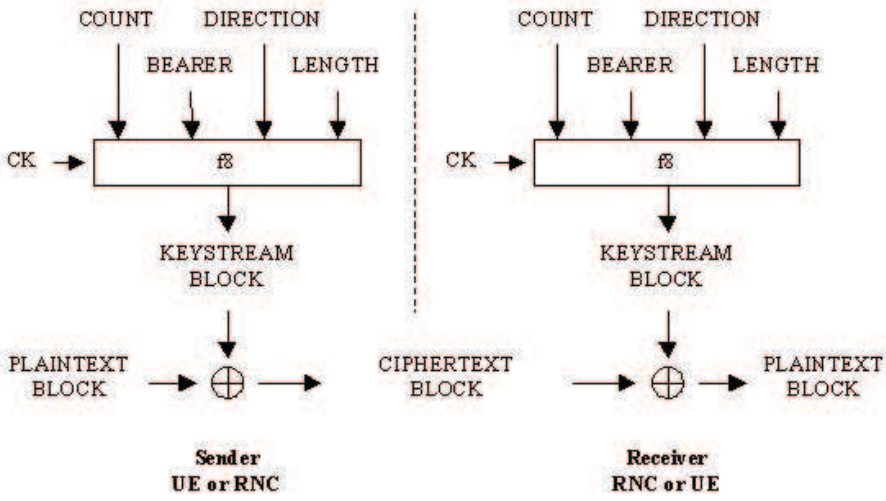

Figure 6.5: Function $f8$ and it's inputs [20].

6.3.3.1 Confidentiality function - $f8$

The $f8$ function, see figure 6.7, is the confidentiality function of UMTS. It's task is to provide confidentiality, of user and signaling data, between the UE and the RNC on the radio access link. $f8$ supports message lengths from 1 to 20000 bits and is based on a combination of the Output feedback mode (OFB) and Counter mode, see details in section 4.4 and 4.3 respectively.

The inputs to f8 are:

BEARER 5 bits identifying the bearer

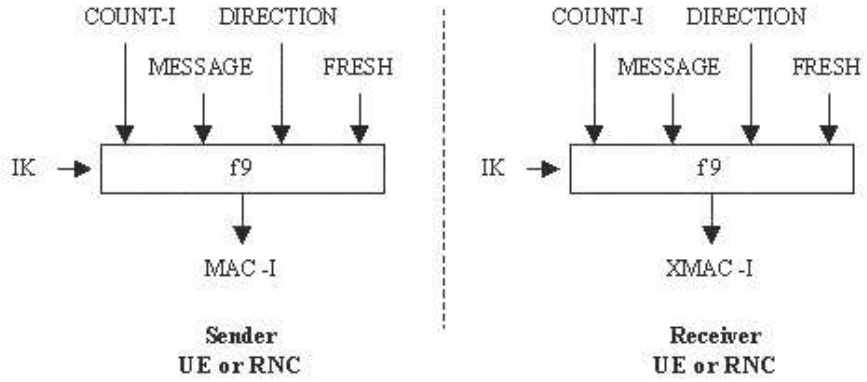

Figure 6.6: Function $f9$ and it's inputs [20].

COUNT 32 bits time dependent input

DIRECTION 1 bit identifying the direction of the transfer (upload/download)

LENGTH The length of the message (plaintext) to be encrypted. The bit size differ, but in the c-code implementation it is 32 bits.

MESSAGE The plaintext to be encrypted, with size LENGTH, max 20000 bits.

KEY The 128 bits key CK derived from the AKA.

Based on these inputs, the $f8$ creates a key stream (KS) which is xor'ed with the message. Because the $COUNT$ will probably never repeat itself while the current CK is used, the key stream will always differ from previously generated key streams. [2]

Creating the key stream: The inputs $BEARER$, $COUNT$ and $DIRECTION$ are concatenated, and zeros are added as padding to create a 128 bit string

$$S = COUNT||BEARER||DIRECTION||0..0 \tag{6.1}$$

Then a CK' is created by xor'ing the Key modifier (KM; every second bit a 0 or 1, beginning width the rightmost bit as a 1) with the CK. Then S and CK' are sent as input to the Kasumi algorithm E.

$$IV = E_{CK'}(S) \tag{6.2}$$

[2] By xor'ing two ciphered messages (CM) one can get the xor of to messages, i.e. $CM_1 \oplus CM_2 = (KS_1 \oplus M_1) \oplus (KS_1 \oplus M_2) = M_2 \oplus M_1$

This IV is then used as an input to the Counter/OFB mode, see figure 6.7 [3]. The counter mode part of $f8$ uses a counter named BLKCNT, that counts the number of blocks we have created up until now. The key stream blocks (KSB_n) is then created by:

$$KSB_n = E_{CK}(IV \oplus BLKCNT \oplus KSB_{n-1}) \tag{6.3}$$

where $KSB_0 = 0..0$. By repeating this operation we create $n = LENGTH/64$ (rounded up to the nearest integer) number of blocks to be xor'ed with the MESSAGE to create the cipher text C.

$$C = KSB_{1 \to n} \oplus MESSAGE \tag{6.4}$$

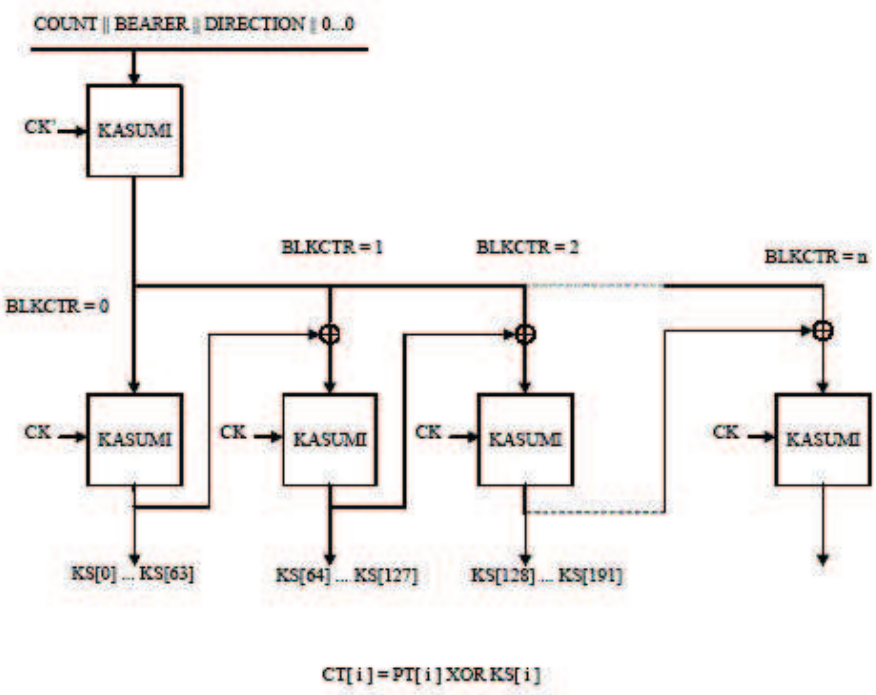

Figure 6.7: Function $f8$ [21].

Properties: The design of the $f8$ has maintained the advantages of the OFB mode, i.e. when transmitting information over an insecure and unstable line (UTMS: air), errors in the transmission are not conveyed to the rest of the message, just this one block of size 64 bits.

[3] By sending the IV through the Kasumi algorithm the designers prevent an attacker from designing an IV that will annul the counter part of the $f8$ function.

6.3.3.2 Integrity function - $f9$

The integrity function of Kasumi, $f9$, verifies the authenticity of the data sent between the UE and the RNC. $f9$ produces a 32 bits Message Authentication Code by using the Kasumi block cipher. The $f9$ is based on a Cipher block chaining MAC, see 4.2, in order to create the fingerprint of the message. The integrity function can handle messages up to 5000 bits.

The inputs to the function f9 are:

FRESH Random number of 32 bits

COUNT-I 32 bits time dependent input

DIRECTION 1 bit identifying the direction of the transfer (upload/download)

LENGTH The length of the message (plaintext) to be encrypted. The bit size differ, but in the c-code implementation it is 32 bits.

MESSAGE The plaintext to be encrypted, with size LENGTH, max 20000 bits.

KEY The 128 bits key IK derived from the AKA.

Creating the MAC First we concatenate parts of the input to a long string, and add a 1 followed by zeros to create a string with length which is a multiple of 64.

$$S = COUNT - I||FRESH||MESSAGE||DIRECTION||1||0..0 \quad (6.5)$$

Then we divide the string S into n parts of 64 bits(S_1, ..., S_n). Each part S_i is xored with S_{i-1} and sent through the Kasumi algorithm (E). The outputs are then xor'ed together to a string of 64 bits. This string is then used as input to E, but now the key IK has been xor'ed with the KM. [4] And leftmost 32 bits from the final calculation is the output from the f9 function.

Properties: The $f9$ function has preserved the main shape of the CBC-MAC, and thus inheriting the advantages. In addition all intermediate outputs (which are pseudo-random) are xored together and sent through one more round of Kasumi. This was done to add some extra security to the $f9$ function. For a thorough security assessment see section 8.4.3.2.

[4] KM here is the flipped version of the f8's KM

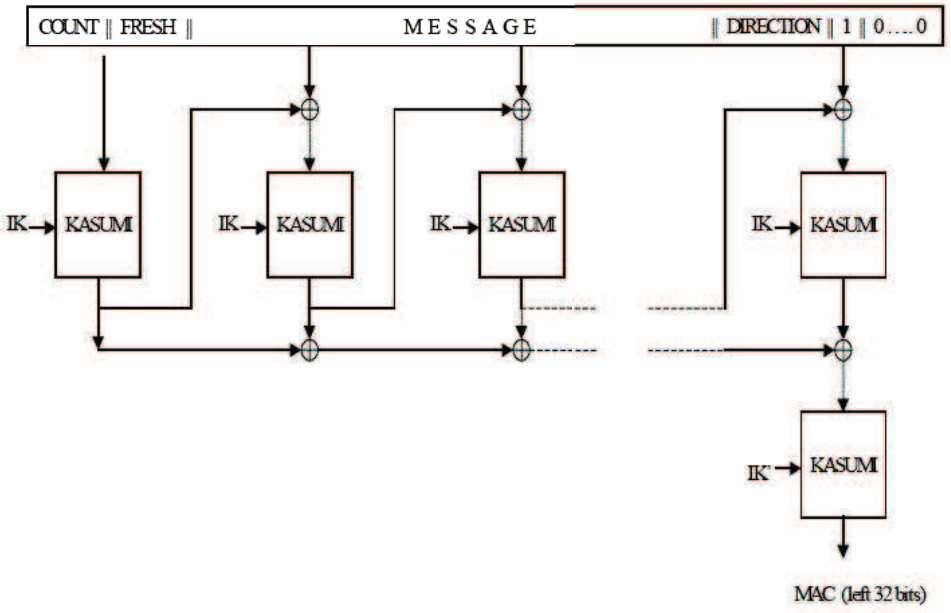

Figure 6.8: Function $f9$ [21].

6.4 Summary

In this chapter we have been given an overview of the UMTS architecture and security architecture. The UMTS network consists of several smaller parts which has a complex collaboration.

The main issues of this chapter are that the confidentiality and authentication runs from the User equipment to the RNC, and that they run both ways. Then the UMTS network knows whom the User Equipment belongs to, and the User/subscriber knows that the current network is authorized by its Home environment/operator.

6.4.1 Conclusion

We have seen the need of security features and mechanisms in UMTS, and we have been shown the functions and protocols needed to implement these. In particular the $f8$ and $f9$ functions, along with Kasumi play a vital part. The analysis of these functions will therefore play a big part in the remaining chapters.

Part II

Block ciphers in UMTS

Chapter 7

Kasumi

In the context of 3GPP, the designers needed a secure algorithm for use in the $f8$ and $f9$ functions. The designers wanted a cipher that was fast and easy to implement in hardware. So, when designing against known attacks, resistance against theoretical attacks were deliberately left out, and the designers "only" wanted the algorithm secure enough.

The Kasumi chapter will try to give the reader a better understanding of how the cipher really works by breaking the cipher up into smaller parts. Then give some implementation aspects. First, we will discuss the background of Kasumi and it's predecessor.

7.1 Designing Kasumi

In the block cipher Misty[22] the designers found a good foundation for Kasumi. Misty is designed to be provable secure against the two most prominent methods of cryptanalysis; linear and differential cryptanalysis (see section 5.2.3 and 5.2.2 respectively). The Misty algorithm is an iterated cipher consisting of several smaller functions which contained the desirable qualities.

The Misty algorithm was originally designed by Mitsuru Matsui to be fast on all types of platforms, hardware or software, not limiting the use of Misty to only one type of machine. Matsui also wanted Misty to be provable secure (see section 8.1.3.1) against linear and differential cryptanalysis.

Provable security was first introduced by Nyberg and Knudsen in [23] where the main idea was to prove small (in input and output) components secure, and then use a Feistel network to build larger block sizes. Matsui was the

first to create an algorithm intended for extended use that actually had adopted these qualities.

7.1.1 From Misty to Kasumi

Nevertheless, the 3GPP designers did not want to adopt Misty without further ado. They did in fact make a few changes. The largest change was done to the key scheduling part, where functions (FL) was replaced by shift operations and by xoring in some constants (See table 7.3). The motivation was found in that the changes lead to less hardware consumption and saved time when generating the keys from the key schedule. A second change made, was the replacement of the substitution table for the $S9$ box. This was also done in order to decrease hardware use and probably save some computational time. The need to increase hardware speed came due to the relocation of the FL function, which made the hardware simpler, but slower[24]. The FL function was also changed with the addition of rotate shift functions, no effect on the hardware, but it was conjectured that cryptanalysis became more difficult.

7.2 Algorithm description

Kasumi is a block cipher built on an eight round Feistel network[25]. The Kasumi algorithm takes two inputs K and P and gives one output C, where $C = KASUMI(P, K)$ The input K is a key with length 128 bit and P, the plaintext to be ciphered, is of length 64 bit. C is also of length 64.

Kasumi is then a family of functions (See section 3.1), indexed by the key K:

$$KASUMI : \{0,1\}^{128} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64} \tag{7.1}$$

7.2.1 The journey through Kasumi

As with other Feistel based algorithms, the input P is split into two halves $L_0||R_0$, each of length 32 bit. ($P = L_0||R_0$) For each Feistel round $i, 1 \leq i \leq 8$, the algorithm computes the following tasks using f as the round function:

$$R_i = L_{i-1} \tag{7.2}$$

$$L_i = R_{i-1} \oplus f_i(L_{i-1}, RK_i) \tag{7.3}$$

The output from the eighth Feistel round is $L_8||R_8$.

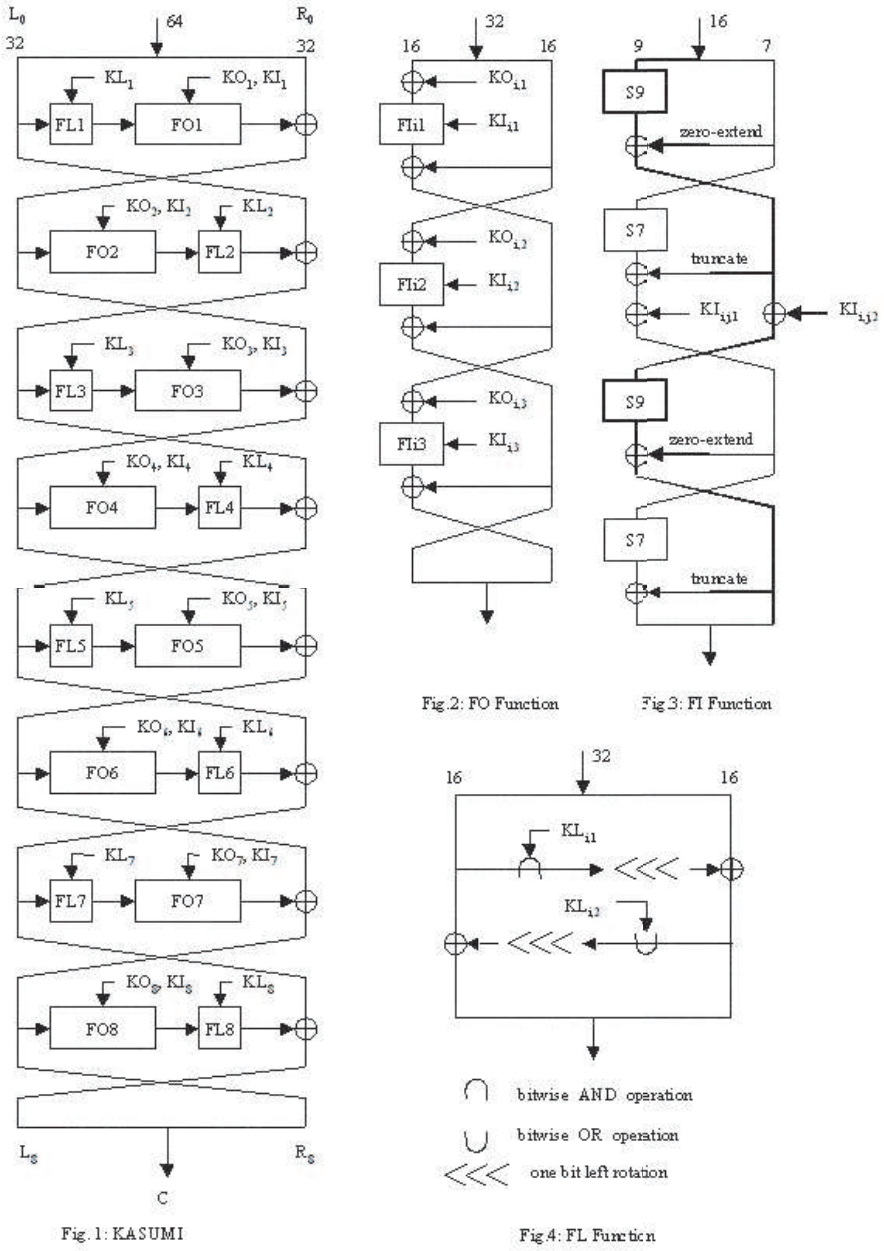

Figure 7.1: The internal of Kasumi[25]

For each Feistel round a unique round key RK is constructed from some of the initial parameters and the algorithm itself (see section 7.3 for more information on key handling).

7.2.2 The round function (f)

The function f, as shown above, takes two inputs, text I and roundkey RK. I is 32 bit long and RK is 128 bit. The round key is then split into three subkeys KI_i, KO_i, KL_i.

Sub key	Bit length	Sub function
KL_i	32	FL
KO_i	48	FO
KI_i	48	FO

Table 7.1: Sub keys and corresponding sub functions

The round function f takes on two different forms, depending on if i is even or odd, with the help from two new subfunctions FL and FO.

Even rounds:

$$f_i(I, RK_i) = FO(FI(I, KL_i), KO_i, KI_i) \tag{7.4}$$

Odd rounds:

$$f_i(I, RK_i) = FI(FO(I, KO_i, KL_i), KL_i) \tag{7.5}$$

7.2.3 FL

The security of Kasumi does not depend on this light weight function. FL is included in Kasumi in order to scramble the output, making each bit harder to track through Kasumi.

The input to FL, is as described above, the key KL_i and the text I. KL_i is split into two halves $KL_i = KL_{i,1}||KL_{i,2}$ and so is the input I. $I = l||r$.

The output, $l'||r'$, from FL is the result from the following equations and is based on the subfunction, ROL, which is the circular shifting of each bit one step to the left:

$$r' = r \oplus ROL(l \cap KL_{i,1}) \tag{7.6}$$

$$l' = l \oplus ROL(r' \cup KL_{i,2}) \tag{7.7}$$

7.2.4 FO

The FO function is the non-linear part of Kasumi, it uses a subfunction FI, see section 7.2.5, and xor's parts of the input with the result from FI. The keys KO_i and KI_i are split into three subkeys, each of length 16 bit, and the input I is split into two $(l_0||r_0)$. The FI subfunction is ran three times as a round function in a modified Feistel network. The modified Feistel network is designed by Matsui [26] and is called a Misty type transformation[22]. For $1 \leq j \leq 3$:

$$r_j = FI(l_{j-1} \oplus KO_{i,j}, KI_{i,j}) \oplus r_{j-1} \tag{7.8}$$

$$l_j = r_{j-1} \tag{7.9}$$

The output from the FO function is the concatenation of $r_3||l_3$.

7.2.5 FI

The inputs are text I and key $KI_{j,i}$. This function also splits up the input, but now in unequal parts. The left part consists of 7 bits and the right of 9 bits. The inputs are then passed on to the substitution boxes (s-box) $S7$ and $S9$.

FI serves as the randomizing part of Kasumi, it uses the Misty type transformation in it's design, with the $S7$ and $S9$ functions as a basis for the FI round function. FI does also consist of two other small functions $ZE(x)$, which adds two zeros to the most significant end of the 7 bit input x, and $TR(x)$, which removes two bits of the most significant end of the 9 bit input x. See also figure 7.1.

$L_1 = R_0$	$R_1 = S9[L_0] \oplus ZE[R_0]$
$L_2 = R_1 \oplus KI_{i,j,2}$	$R_1 = S7[L_1] \oplus TR[R_1] \oplus KI_{i,j,1}$
$L_3 = R_2$	$R_3 = S9[L_3] \oplus ZE[R_3]$
$L_4 = S7[L_3] \oplus TR[R_3]$	$R_4 = R_3$

Table 7.2: FI function

7.2.5.1 S7 - S9

These two s-boxes take, correspondingly, 7 and 9 bits as input, and they have a one-to-one relation with the output. $S7(\{0,1\}^7) \rightarrow \{0,1\}^7$. This mapping is designed in such a way that it is easy to implement either as a lookup table or by using gate logic. See the appendix for the lookup table.

The s-boxes of Kasumi, how small and insignificant they may seem, is the foundation of the security of Kasumi. It is therefore vital for the s-boxes to be as secure as possible (See section 8.1.3.1 for more details) in order for the whole of Kasumi to be as secure as possible.

7.3 Key schedule

As mentioned before, the individual keys used in the different functions are derived from the original 128 bit key K. Each bit of the 128 bit key is used once and only once for each round, and in addition each bit is used in a different way for each round[21]. K, given as input to Kasumi, is divided into 8 smaller parts, each of size 16 bit (see equation 7.10).

$$K = K1||K2||...||K8 \tag{7.10}$$

Then a second array of keys K' is derived from the keys in equation 7.10 (for $1 \leq j \leq 8$):

$$Kj' = Kj \oplus Cj \tag{7.11}$$

where the constants $C1 \rightarrow C8$ are defined in table 7.3 and written in hexadecimal. The use of the constants prevents chosen plaintext attacks that are more effective than exhaustive search. This is due to the fact that the relations between the keys, in two rounds r and $r+1$, are not fixed, resulting in that key bits becomes more difficult to track through the rounds[21].

C1	0123
C2	4567
C3	89AB
C4	CDEF
C5	FEDC
C6	BA98
C7	7654
C8	3210

Table 7.3: Key scheduling constants in hexadecimal[25]

All subkeys are derived from K and K' and can be found in figure 7.2, where $<<< n$ is the left circulation of n bits.

7.4 Implementation aspects

We recall that one of Matsui's intentions was to make Misty fast both in hardware and software, and as a result Matsui developed the Misty-type-

	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6	Round 7	Round 8
$KL_{i,1}$	K1<<<1	K2<<<1	K3<<<1	K4<<<1	K5<<<1	K6<<<1	K7<<<1	K8<<<1
$KL_{i,2}$	K3′	K4′	K5′	K6′	K7′	K8′	K1′	K2′
$KO_{i,1}$	K2<<<5	K3<<<5	K4<<<5	K5<<<5	K6<<<5	K7<<<5	K8<<<5	K1<<<5
$KO_{i,2}$	K6<<<8	K7<<<8	K8<<<8	K1<<<8	K2<<<8	K3<<<8	K4<<<8	K5<<<8
$KO_{i,3}$	K7<<<13	K8<<<13	K1<<<13	K2<<<13	K3<<<13	K4<<<13	K5<<<13	K6<<<13
$KI_{i,1}$	K5′	K6′	K7′	K8′	K1′	K2′	K3′	K4′
$KI_{i,2}$	K4′	K5′	K6′	K7′	K8′	K1′	K2′	K3′
$KI_{i,3}$	K8′	K1′	K2′	K3′	K4′	K5′	K6′	K7′

Figure 7.2: The Subkeys of Kasumi[25]

transformation instead of using the well used and documented Feistel network. He has also assessed which types of operations that provides both safety and speed, his conclusion was that logical operations (AND, OR, XOR) and lookup tables provides some of both properties given above. However, not all optimizations are available both in hardware and software, some small differences to the implementation of Kasumi may exists.

7.4.1 Parallel computing

When Kasumi is implemented in hardware or software it is possible to run some functions in parallel, and thereby decreasing the calculation time. In FI it is possible to run the first and the last $S7$ and $S9$ together in parallel and two consecutive rounds of the FI in the FO function[24](See figure 7.3).

7.4.2 Lookup tables

The functions $S9$ and $S7$ are possible to implement as a lookup table/array. However, the efficiency of a lookup table may vary from computer to computer depending on the memory access speed[22]. In hardware it is possible to stream line the lookup table using logic gates, which removes the delay when accessing the computers memory[22, 24]

Lookup table example: $S7[1] = 54$ and $S7^{-1}[54] = 1$

7.5 Summary

In this chapter we have seen how the Kasumi algorithm was designed from the Misty cipher as an iterated cipher using a Feistel network. Kasumi

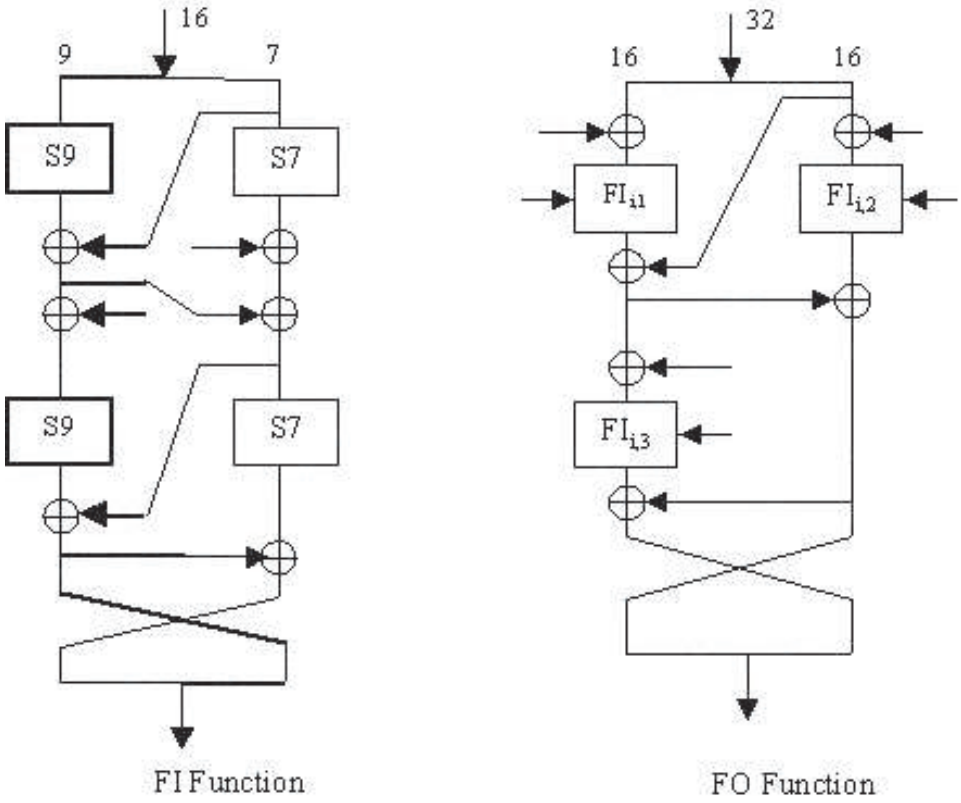

Figure 7.3: Optimization of FI and FO in Kasumi[24]

consists of several smaller parts with the s-boxes $S7$ and $S9$ as the smallest. These parts create a 64 bits block cipher, with a 128 bit key creating the 64 bit output of Kasumi.

Both Misty and Kasumi are built on the basis of the theory of provable security against differential and linear attacks. And should therefore be "immune" to these attacks.

Chapter 8

Security analysis of Kasumi and UMTS' encryption and integrity schemes

Security is all about trust. Who and what do we trust? Scientists have for a long time searched for the ONE notion of security, the absolute proof of security that operates without trust. Unfortunately no such notion exists (today) that can be applied in practice. Therefore we have to take a look at the best possible proofs of security and resilience against known attacks existing today.

This chapter will give an overview of what kind of principles and results Kasumi and the UMTS operation modes are built upon. The proper amount of background theory (See section 3) is given in order for the reader to understand the types of "proofs" that are available to deem UMTS' encryption ($f8$) and integrity ($f9$) functions strong enough to endure the next 20 years. In addition there will be a large section concerning possible attacks on Kasumi and the UMTS schemes.

8.1 Security of Kasumi and its building blocks

The users of UMTS need to know whether they can trust the $f8$ and $f9$ functions. In order to give them an answer we need to take a closer look at the security these functions provide.

The cornerstone of both the $f8$ and $f9$ functions is the Kasumi block cipher, and in order for the functions to be secure, the Kasumi also needs to be secure. Therefore this section will first concern about the security of Kasumi;

the design principles, the smaller building blocks of Kasumi and then how the whole Kasumi, based on the smaller parts, can prove itself worthy.

8.1.1 General description of Kasumi

We recall from Chapter 7 that Kasumi is built on an eight round Feistel network, where each round consists of two functions (FL and FO). FO is also built as a modified Feistel network (3 rounds) with the FI as the round function. This modified Feistel network is called by [5], a *Misty-type transformation*. FI consists of two s-boxes (S7 and S9) in a four round Misty type transformation.

8.1.2 Rationale of Feistel networks

The main advantage of using Feistel networks (See 2.3.2 for description) is that the round function f doesn't need to be invertible [1], you only have to invert the order of the round keys.

More important for the security is that each round of the Feistel network is a permutation of the original input, making the whole Feistel network one big permutation. Feistel networks are very important when proving security against linear and differential cryptanalysis. This will be discussed later in this chapter.

8.1.3 The s-boxes

We recall that Kasumi uses two s-boxes, one which takes 7 bits of input ($S7$) and the second that takes 9 bits ($S9$). Both boxes can be thought of as a lookup table, where each input corresponds to the same amount of output bits. The s-boxes are defined as bijective[1] mappings of one Galois Field(GF)[2] to another; $f : GF(2^n) \rightarrow GF(2^n)$.

The s-boxes are the smallest components of Kasumi, which upon the whole security against linear and differential cryptanalysis lies. We will therefore in this section discuss the background and rationale of the s-boxes any why they were designed as they are. However, first we need to explain how we can provide security, as there are two main approaches for a block cipher to be secure. These two properties come in addition to the general requirements, of section 2.2.2.

[1] one-to-one correspondence

[2] Given a prime p and a natural number n it is possible to create a field with p^n elements

The first and most obvious way is to show that no known attack existing today is able to *break* your cipher, by actually mounting an attack of each kind at the block cipher, or proving specific resistance against an attack. However, resilience against one attack does not protect against other attacks. The second way is for the cipher's output to appear totally random, undistinguishable in polynomial time from a random function/permutation (See section 3.2 for more details).

For the remainder of this chapter we will take Kasumi through both these two approaches. First by showing that Kasumi is provable secure against linear and differential cryptanalysis, then by giving an overview over the best possible attacks against Kasumi and finally showing that Kasumi does behave as a pseudo random permutation.

8.1.3.1 Provable security of s-boxes

The notion of provable security of DES like ciphers against differential cryptanalysis was defined by Nyberg and Knudsen in [23] as the property of a function F having a small enough probability for differentials and linear structures to occur. The definitions of linear (LP) and differential (DP) probability are found in [23, 26]:

Definition 3. *Let $F_k(x)$ be the function F with input x and k. x having length n and k having length l.*

$$DP^F \stackrel{def}{=} \frac{1}{2^l} \sum_k \max_{\Delta x \neq 0, \Delta y} \frac{\#\{x | F_k(x) \oplus F_k(x \oplus \Delta x) = \Delta y\}}{2^n} \tag{8.1}$$

Here we are counting the number of x's that yield the same output difference Δy when used in $F_k(x) \oplus F_k(x \oplus \Delta x)$ for a given difference Δx. If two or more x's derives the same Δy, the number of possible used values for Δy decreases. The goal is that no two x yields the same Δy (Perfect nonlinear, see below).

$$LP^F \stackrel{def}{=} \frac{1}{2^l} \sum_k \max_{\Gamma x \neq 0, \Gamma y} (2 \frac{\#\{x | x \bullet \Gamma x = F_k(x) \bullet \Gamma y\}}{2^n} - 1)^2 \tag{8.2}$$

Where $x \bullet y$ is the parity of the bitwise product (x AND y). Γx and Γy represent the approximation.

The numerator is counting the number of successful approximations (Γx,Γy), between the input x to the function F and the output $F_k(x)$ for each key k. The numerator will be a number between 0, when no approximation has been found, and 2^n when all approximations succeeds. However, most counts will have a value of approximately 2^{n-1}. If the numerator has a count that differs

from 2^{n-1} LP grows larger. This is what we want to avoid when designing a function with resistance against Linear cryptanalysis.

The rest of the calculations of LP ($2(\frac{\cdots}{2^n} - 1)^2$) are just to transform the count to a number between 0 and 1.

Almost perfect nonlinear permutations Now that we have defined how to calculate the probabilities, we can introduce a notion defined by Nyberg and Knudsen in [23], *almost perfect nonlinear permutations*, but first we will define *perfect nonlinear functions*.

The lowest possible probability (perfect nonlinear) for DP, of equation 8.1 (and LP), appears when every x gives a unique Δy in the $F_k(x \oplus \Delta x) = \Delta y$. This probability is calculated in equation 8.3, where $|k| = 2^l$:

$$DP^F = \frac{1}{2^l} * 2^l * \frac{1}{2^n} = 2^{-n} \tag{8.3}$$

However, the Kasumi s-boxes are not functions, but permutations and cannot obtain lesser LP and DP than 2^{1-n} [23]. Permutations obtaining this LP and DP is called *almost perfect nonlinear*.

The reason why permutations cannot be perfect nonlinear is that they only obtain half of the possible values in the definition of DP. Take a closer look at the equation 8.1. We are trying to count the number of x's that gives a specified output Δy. There are at least two, the number depends on the permutation, x values that yields the same Δy in $F_k(x \oplus \Delta x) \oplus F_k(x)$. This is due to the fact that when Δx is fixed and x varies, then x will eventually be equal to $x \oplus \Delta x$ which yields:

$$F_k((x \oplus \Delta x) \oplus \Delta x) \oplus F_k(x \oplus \Delta x) = F_k(x) \oplus F_k(x \oplus \Delta x) \tag{8.4}$$

Therefore two different x values give the same Δy and the equation of 8.3 now obtains twice the probability:

$$DP^F = \frac{1}{2^l} * 2^l * \frac{2}{2^n} = 2^{1-n} \tag{8.5}$$

In here lies also the reason to why the s-boxes have uneven numbers (7 and 9, not 8 and 8). This is because odd numbers in a bijective function has a possible minimal LP and DP of 2^{1-n} (see equation 8.5), while it is conjectured that when the functions have even numbers the minimal LP and DP are 2^{2-n}[22]. Therefore by choosing an odd number for your s-box you are getting the lowest possible linear and differential probability.

8.1.3.2 S7

The S7 box is chosen in such a way that the corresponding input and output bits are maximally non-linear (almost perfect nonlinear), for a permutation.

To find functions for $GF(2^7)$ that are maximally nonlinear we turn to *Kasami exponents*[21] (not to be confused with Kasumi). Kasami found a set of exponents that give the function of $GF(2^7)$ maximal nonlinear properties. The function chosen for Kasumi's $S7$ box is x^{81}.

To verify the maximal nonlinearity of $S7$ we will check whether 81 of x^{81} is a Kasami exponent. The calculation and theorem are retrieved from the "3GPP Kasumi Evaluation Report" section 9.2.1.3.1 [21].
Theorem 1. *A linear transformation x^d over $GF(2^n)$ has optimal nonlinear probabilities if for $n = 2m + 1$ such that $2 \geq k \geq m$, $gcd(k, m) = 1$ and $d = 2^{2k} - 2^k + 1 (mod\ 2^n - 1)$.*

Two exponents are equivalent if $d' = 2^t d$.

In our case $n = 7, d' = 81$. k is either 2 or 3, and if we factorize 81 to

$$81 = 80 + 1 = 64 + 16 + 1 = 2^6 + 2^4 + 1 \tag{8.6}$$

and uses that two exponents can be equivalent, we see that

$$d' = 81 = 2^6 + 2^4 + 1 = 2^8 - 2^6 + 2^4 (mod\ 127) = 2^4(2^4 - 2^2 + 1)(mod\ 2^7 - 1) \tag{8.7}$$

d is then $2^4 - 2^2 + 1 (mod\ 2^7 - 1)$ which gives $k = 2$. The $gcd(2, 3) = 1$ and all the conditions of Theorem 1 are fulfilled.

Through this calculation we have verified that the $S7$ box of Kasumi is a almost perfect linear permutation, setting $S7$'s DP and LP to 2^{-6}.

It is also proven that it is virtually impossible to create a probabilistic approximation of the S7 box [21].

8.1.3.3 S9

As $S9$ is a composition of x^5 and a linear output transformation over $GF(2^9)$ the Kasami exponents are not valid here, and no calculation is available. However, the S9 box does provide maximal non-linearity ($LP = DP = 2^{-8}$)[21]. It contains 511 linear structures. To compensate for this small weakness the S9 is constructed in such a way that this property is not passed on to the FI and FO functions [21].

8.1.4 FI and FO functions

In 1993 Nyberg and Knudsen proved that small, secure (differential and linear resistant) components can be merged by a Feistel network in order to create larger, secure block sizes [23]. This theorem is denoted as Theorem 2.

Theorem 2. *It is assumed that in a DES-like cipher*[3] *with* $f : GF(2)^m \rightarrow GF(2)^n$ *the inputs to* f *at each round are independent and uniformly random. Then the probability* p *of an r-round differential,* $r \geq 4$*, is less than or equal to* q^2*. Where q=LP or DP.*

Originally Nyberg and Knudsen proved p to be $2q^2$, but later Aoki and Ohta improved this limit to q^2[27].

It is this theorem that Matsui has taken advantage of when designing Misty (Kasumi's predecessor). The s-boxes have small differential/linear probabilities [21, 22], and can thus be used in the way that Nyberg and Knudsen suggested.

In [26] Matsui adapts Theorem 2 to also cover two other types of networks (Misty type transformations) different than the Feistel network, but keeping the probability to q^2. These two other networks are used in FI and FO [22].

So what Matsui does is using Theorem 2 and his modified Feistel networks, three times, in order to compose the larger Kasumi from the small s-boxes. First by concatenating the two s-boxes in the FI function (7+9=16 bits), then in the FO function where three FI functions are added to create 32 bits, and then finally in the outer Feistel network of Kasumi where 8 rounds of FO creates a 64 bits permutation.

What the theorem from Nyberg and Knudsen tells us is that it suffices to prove the small components secure against differential/linear attacks for the whole Kasumi function to be differential/linear secure.

By knowing the linear (LP) and differential(DP) probabilities of the s-boxes and using Theorem 2 we can calculate the LP and DP of FI, FO and Kasumi.

$$DP_{FI} = LP_{FI} = 2^{-8} * 2^{-6} = 2^{-14} \tag{8.8}$$

$$DP_{FO} = LP_{FO} = (2^{-8} * 2^{-6})^2 = 2^{-28} \tag{8.9}$$

$$DP_{KASUMI} = LP_{KASUMI} = ((2^{-8} * 2^{-6})^2)^2 = 2^{-56} \tag{8.10}$$

[3] DES-like meaning built on a Feistel network

In comparison we can calculate the DP and LP for a Kasumi constructed variant using two s-boxes defined over GF(2^8): DP and LP are then calculated to $(((2^{-8+2})^2)^2)^2 = 2^{-48}$. Which is considerably larger than DP_{KASUMI}.

8.1.5 Complexity of linear and differential attacks

Now that we have calculated the probabilities of a possible differential/linear structure to occur (see equation 8.10), we can find the time and data complexity of an attack on Kasumi using these structures.

8.1.5.1 Differential complexity

The relationship between differential probabilities and the lower bound of the complexity can be found in [9] Theorem 1. The simplified version is given as Theorem 3

Theorem 3. *Let $Comp(r)$ denote the lower bound on the complexity of a differential cryptanalysis of an r-round iterated cipher, which following [11], is defined as the number of encryptions used.*

$Comp(r) \geq \frac{2}{DP - \frac{1}{2^m - 1}}$,

where DP is the equation of 8.1 and m is the block length of the cipher.

Using theorem 3, we can calculate the number of encryptions needed to launch a differential attack on a full 8 round Kasumi.

$$Comp_{Kasumi}(8) \geq \frac{2}{2^{-56} - \frac{1}{2^{64}-1}} \geq 2^{57} \tag{8.11}$$

I would like to emphasize that equation 8.11 only gives the lowest possible bound, the actual required number of chosen plaintexts may be substantially higher. Only a more thorough investigation can reveal the true number.

In comparison to the real number of chosen plaintexts needed to "break" Kasumi, the authors of [9] has conducted a complete investigation of PES which has a $DP_{PES} = 2^{-58}$. Where the true number of chosen plaintext was calculated to be 2^{64}, while Theorem 3 only gives a lower bound of 2^{59}[9].

It is therefore highly likely that a linear attack on Kasumi requires more than 2^{57} chosen plaintexts.

8.1.5.2 Linear complexity

Adopting the method used by Matsui in his paper *Linear Cryptanalysis Method of DES Cipher*[12] we can calculate the required number of known

plaintexts N needed to conduct a linear attack on Kasumi. Where p is the probability of a linear structure with a deviation furthest away from $\frac{1}{2}$.

$$N = |p - \frac{1}{2}|^{-2} = |(\frac{1}{2} + LP) - \frac{1}{2}|^{-2} = |LP|^{-2} = 2^{112} \tag{8.12}$$

Even though this formula gives $N = 2^{112}$ it is not necessary to find them all[4]. This is due to the fact that when we reach 2^{64} known plaintexts we have a complete lookup table for Kasumi with the current key, and we do not need the key to decipher any messages.

However, to find all 2^{64} known plaintexts, even outside the 3GPP environment, is virtually impossible. So the provable security of linear cryptanalysis leads to a complexity of the attack which is impossible to handle.

8.1.6 Summary

This section has given a thorough review of how small provable secure s-boxes can be fitted together to create larger functions which preserve their provable security. Secondly, we have shown how to transform the provable security of differential and linear probability into computational complexity.

Kasumi, with it's provable security, has been shown to have a complexity, of regular differential and linear attacks, out of bounds for today's polynomially bound computers. Kasumi can therefore be reckoned as secure against these types of attacks.

Remarks The same conclusion was also drawn by Alex Biryukov in his *Block Ciphers and Stream Ciphers: The State of the Art*[3] where he states on page 12, and I quote without his references: *At the present moment there is no known weaknesses in this cipher in-spite of considerable research efforts around its predecessor MISTY1 and on KASUMI itself.*

8.2 Cryptanalysis of Kasumi

Due to the fact that we have no strong proof of security of block ciphers, a block ciphers gains trust, among the crypt-engineers, proportionally with the amount of analysis done to the block cipher. Kasumi (and it's predecessor Misty), even though it is a young block cipher, has undergone a quite substantial amount of cryptanalysis. In this section I will provide an overview of the most significant attacks on Kasumi.

[4]Recall that Kasumi takes 64 bit input and yields 64 bit output. There are "only" 2^{64} different possible plaintexts and ciphertexts

8.2.1 Differential cryptanalysis

Although Kasumi was designed to provide provable security from differential cryptanalysis (See section 5.2.2), there still exists some more sophisticated differential attacks on a reduced-round Kasumi.

8.2.1.1 Impossible differentials attack on Kasumi

In the 3GPP Kasumi Evaluation Report from SAGE [21] it is stated that the FI functions does not contain any impossible differentials (see section 5.2.2.1), while the FO function does. These differentials are transferred to a 2 or 3 round Kasumi *without* the FL function. Adding the FI function most of the impossible differentials are wiped out, and they become key-dependent.

Best known impossible differential attack The best known attack using impossible differentials are against a 6 round Kasumi using $2^{53.3}$ chosen plaintexts and using 2^{100} encryptions, exploiting the impossible differential and the structure of the FO function[21].

No attack on the full 8 round Kasumi has been found using impossible differentials.

8.2.1.2 Best known xor-restricted related-key attack (RKA) on Kasumi

An RKA (see section 5.2.2.2) attack on a six round Kasumi (reduced) requires a chosen plaintext attack with $3 * 2^{17}$ plaintext pairs (X and X') and a maximum of 2^{112} trials to find the current key.

The attacks described in [28], are the best xor-restricted related-key attacks known, yet they pose no real threat against the security of Kasumi. This is due to the fact that they rely on a reduced round Kasumi (5 or 6 rounds) and the use of a special type of related keys.

These type of related keys are, accordingly to the authors, regarded as impossible to implement, due to the 3GPP environment.

8.2.1.3 Higher order differential attack on Kasumi

Kasumi's predecessor Misty has undergone much analysis using 6^{th} and 7^{th} order differentials[21] (see section 5.2.2.3 for more details). The best known

attack to Misty, given in [21] is an attack using 1408 chosen plaintexts on a 5 round Misty by Tanaka et al. in *Strength of MISTY1 without Fl function for Higher Order differential Attack*. However, this attack is not transferable to Kasumi [21], due to the fact that the FI function is modified. No higher order differential has neither been found for the full 8 round Kasumi/Misty[21].

8.2.2 Linear cryptanalysis against Kasumi

As mentioned, Kasumi is designed with provable security against linear cryptanalysis (section 5.2.3 contains more information). However, it may be possible to increase the probability for LP_{FI} (See definition 8.2 in 8.1.3.1) to 2^{-12} by exploring some specific keys[21], lessening the amount of complexity and workload.

8.2.2.1 Best known linear attack

Using these keys it is possible to mount a five round attack on Kasumi using 2^{58} known plaintexts and doing 2^{95} calculations, but this attack can only be used for a fraction of 2^{-3} of the keyspace.

8.2.3 Side channel attacks on Kasumi

According to [21], Kasumi has undergone analysis against timing attacks, simple power analysis and differential power analysis without finding any significant vulnerabilities. (See section 5.2.4 for more details against the individual attacks.)

8.2.4 Statistical evaluation

Although it was proven by Iwata and Kurosawa in [29], that the integrity ($f9$) and confidentiality ($f8$) functions of UMTS was indistinguishable from a random permutation, there was conducted a series of statistical evaluations both on the Kasumi algorithm and the $f8/f9$ functions to verify this proof. The tests on Kasumi were performed on the whole function, FI, FO and on the two s-boxes[21].

The results confirmed that the s-boxes indeed were almost perfect nonlinear, and for FO and FI no linear structures where found. The bit sequences generated with Kasumi did not deviate from random behavior[21]. Neither did the bit sequence generated by $f8$. The $f9$ function did also pass all

dependency tests (That every bit of the output depends on every bit of the key and input).

8.2.5 Conclusion

No known attack is able to break the full 8 round Kasumi. Still the best possible attack is the exhaustive key search. Nevertheless, one cannot disregard that a new unknown attack is able to break Kasumi in the future. However it is appropriate to conclude that Kasumi is safe enough for the next 20 years, due to the fact that no known successful attack against Kasumi exists today.

8.3 Pseudo-randomness of Kasumi

While the chapter 3 gave the theory of pseudo-randomness, this section we will apply this theory to Kasumi to see if Kasumi behaves as a family of pseudo-random permutations or not.

The proof for Kasumi's pseudo-randomness is found in [5], where the authors provide a rigid proof, using the adaptive distinguisher model, of that a four (or more) round Kasumi is a 64 bit pseudo-random permutation (PRP), while the three round is not. See [5] for complete proof. In this section I will give you an outline (short version) of this proof.

8.3.1 Outline of proof

As mentioned above, Kasumi is a 64 bits permutation consisting of 2 main functions FL and FO. The FL function is omitted in the proof, due to the fact that this function is only a linear round key mixing. FO still remains, and it is a 32 bits permutation. FO's security relies on the two s-boxes used in the 16 bits permutation of FI .

In [30] Luby and Rackoff proved that a DES-type transformation with three rounds, with a n-bit pseudo-random permutation as the round function, generates a $2n$-bit permutation. However, Matsui created his own Misty-type transformation (used in FO and FI) so the proof of Luby and Rackoff cannot automatically be applied to the whole of Kasumi. Further, Sakurai and Zheng in [31] showed that the three round Misty network used in the FO function is *not* a PRP, while the four round, used in FI, is [32, 33].

The authors of [5] wanted first to verify that the four round Misty-type transformation and the two s-boxes together created a PRP. In order to find

the distinguishers advantage, the probability of a bad event was derived. By bad we mean the functions (S9 and S7) having equal differences for different outputs. The result of the advantage calculation showed that it was negligible. FI was indeed a (9+7) bit PRP.

Unable to prove the FO a PRP, due to Sakurai and Zheng in [31], Yi et al. had to use the same procedure with the four round Kasumi as with the FI function. They could not apply the Luby and Rackoff theorem, and used only the fact that FI was a PRP. Each round of their simplified Kasumi then contained three FI functions. After calculating the bad events, the results showed that Kasumi was a $4n$ bit PRP. Where $n = 16$ bits of the FI function.

8.4 Security of 3GPP encryption and integrity schemes

8.4.1 On the construction of f8

As mentioned earlier the design of $f8$ is based on the OFB operation mode (see section 4.4), however slightly modified. So in addition to having the properties of an regular OFB scheme the $f8$ mode has a pre-computation using the kernel function of the inputs to the scheme (see section 6.3.3.1, and figure 6.5 for more detailed information). This pre-computation gives two advantages[21]:

Protection against chosen plaintext With the initial round of Kasumi, the inputs to the normal OFB, are not longer known. This makes it more difficult for the chosen plaintext attack to be carried out. This is due to the fact that the inputs are sent through one round of Kasumi with the use of the unknown key.

Protection against collision attacks For a $f8$ construction $f8'$ without the initial Kasumi round, it is possible to distinguish $f8'$ from a random generator. We are able to distinguish between $f8'$ and a random generator if the probability for a collision is "large," however for the $f8$ this probability is low enough[21]. A collision is under the notion of birthday attacks[5]

[5]A birthday attack comes from the surprising result that in a class of 23 pupils, the probability that two persons have the same birthday is greater that 50%. This is used when searching for two different plaintexts having the same fingerprint from a MAC. This is called a collision.

8.4.2 On the construction of f9

As discussed in 6.3.3.2 the construction of $f9$ is built as a modified CBC-MAC mode. With standard CBC-MAC a birthday attack[5] would be possible to implement using 2^{33} messages, but with the $f9$ construction one needs 2^{65} messages. This number of messages is out of reach. $f9$ has therefore gained a significant improvement without getting any additional weaknesses.

8.4.3 Proving the security

In 1997, Bellare et al. showed in [34] that an encryption scheme or a mode of operation (CBC, CTR) is *secure*, within and upper and lower bound of calculations, if the encrypted block cannot be separated from a random permutation from an oracle (left-or-right indistinguishability (LOR)). Given that the underlying primitive, i.e. the block cipher in use, was a pseudo-random permutation (PRP).

The term *secure* is given in the context of the notion of *left-or-right indistinguishability*. LOR is similar to the Adaptive distinguisher model, but int LOR the adversary queries the oracle with two inputs (x_0,x_1). The oracle then replies with either $E_k(x_0)$ or $E_k(x_1)$. Where E is an encryption scheme and k is the currently chosen key. The encryption scheme is regarded as secure if the adversary cannot obtain significant advantage in distinguishing between the left and the right output.

It was under this assumption Yi et al. "proved" the $f8$ function secure in [5]. This proof was later proven to be false by the authors of [35, 29], where they showed that it was impossible to prove whether $f8$ or $f9$ were secure under the assumption that Kasumi was a PRP. To show the proofs false, Iwata and Kurosawa constructed a PRP F with the property that for a constant C and *any* key K, $F_K(x) = F^{-1}_{K \oplus C}(x)$. By applying F to $f8$ and $f9$, the schemes are easily broken with a probability of 100%. The authors stress that their results do not show that $f8$ and $f9$ are insecure using Kasumi, but only that the provable security of $f8$ and $f9$ are nonexistent under the PRP assumption.

Recently (January 2004) Iwata and Kurosawa did in fact prove in [29]that the $f8$ and $f9$ functions were secure, but now under the assumption that the underlying cipher was secure against xor-restricted related-key attacks.

8.4.3.1 Security of $f8$

As the authors of [35] proved, it was not enough for the underlying primitive (Kasumi) to be a PRP. So Iwata and Kurosawa wanted to see if there

existed any way to prove $f8$ and $f9$ secure under the notion of the Adaptive distinguisher model. Because of the $f8$'s use of the key-modifier (See section 6.3.3.1), the minimum assumption was to see whether $f8$ was secure against xor-restricted related-key attacks (See section 8.2.1.2 for more details).

To prove the security of the $f8$ encryption scheme against RKA's, Iwata and Kurosawa designed a weaker, but more flexible version of $f8$. This version ($f8$') made the adversary's job a bit easier, but still, the advantage the distinguisher had, when using the "Adaptive distinguisher model", were negligible. So if the $f8$ function is combined with a RKA resistant primitive the whole scheme is secure. Since the best possible RKA attack on Kasumi pose no real threat against a full 8 round, 3GPP's encryption scheme $f8$ can be regarded as secure.

8.4.3.2 Security of $f9$

In [36] the authors proved that if the underlying block cipher was a PRP and this cipher was used in a CBC-MAC (See 4.2) the property of the PRP was preserved by the CBC-MAC and transformed into a PRF. Further the authors proved that if the MAC was a secure PRF, then the whole scheme is unforgeable. The PRP property of Kasumi was proven by [5] and the $f9$ integrity scheme is based on a CBC-MAC. (See sub-section 6.3.3.2)

But Iwata et al., the authors of [29], wanted to prove that the $f9$ scheme was indeed a PRF. Again they used that the Kasumi cipher was secure against xor-restricted related-key attacks and showed that the advantage the distinguisher had was small by definition and therefore negligible.

8.4.4 Cryptanalysis of $f9$

Lars Knudsen and Chris Mitchell have in [37] described three types of attacks on the 3GPP-MAC-scheme $f9$. These three types are chosen MAC, known MAC and the last uses MAC verifications. Several attacks emerge from each type and their goal is to recover the key or to forge a MAC.

In this section you will find the outlines of these attacks including their complexity.

Throughout this section 8.4.4, m is the block length of the cipher used, and n is the length of the MAC.

8.4.4.1 Exhaustive search

The exhaustive search is a known MAC attack. The attack uses very few known MACs, due to the fact that the $f9$ scheme uses two keys in the calculation of a MAC, K and $K' = K \oplus KM$ where KM is a Key Modifier (See section 6.3.3.2 for more details.), and thus lessening the probability for a collision.

The main idea is that you know some plaintexts and their MACs, then you run the plaintexts through $f9$ using all possible keys (K, K') to see if you can find a match between the derived MAC and your original. If you find a match, it can be the right key k, or you have found a collision.

The probability of a collision is 2^{-m}, where m is equal to 32. The reason for this is that after the final round of Kasumi in $f9$, the 32 rightmost bits are truncated. Therefore 2^{32} blocks will have the same 32 leftmost bits, while the 32 rightmost differ. To rule out those keys that only yields a collision we need to check the key k with other known MACs. The process is continued until k has been found.

This attack has a complexity of $(N + 3) * 2^{|K|} = (N + 3)2^{128}$ encryptions, where N is the number of 64 bits blocks of the plaintext, and 3 because; one Kasumi encryption at the end using K' and two for key derivations[37].

The exhaustive attack is easy to implement, but 2^{128} computations of Kasumi is not possible to accomplish on today's polynomial bounded computers.

8.4.4.2 A forgery attack

A forgery attack is a chosen MAC attack where it is possible to forge a message into having the same MAC as another message. In so doing it is possible to change the content of the message, but (without knowing the key) keeps the MAC derived from the original message.

In order to conduct this attack you have to collect enough messages and their corresponding MACs to find a collision. For the $f9$ function this number q is $\sqrt{2} * 2^{n/2}$[37]. For $f9$ and Kasumi $n = 64$ (block size) and $q = \sqrt{2} * 2^{32}$. When you have found the two colliding messages M and M', construct $2^{n/2}$ messages $M(i) = M|Rand(i)$ and $2^{n/2}$ messages $M'(j) = M'|Rand'(j)$ and their MACs. Where $Rand(i)$ and $Rand'(j)$ are randomly chosen n-bits blocks.

Again one expects to find a collision among these newly constructed MACs. Then take their $Rand(i)$ and $Rand'(j)$ from this collision and xor them

together, the result is denoted Δ. Then every new n-bit block Z, $M(i)|Z$ and $M'(j)|Z \oplus \Delta$ will have the same MAC[37].

The complexity for implementing this forgery attack on $f9$ is $2^{(3n/2)-m+1} = 2^{67}$ different messages, which under the 3GPP environment are out of bounds within the use of one IK/CK (See chapter 6.3.2.5).

8.4.4.3 A key recovery attack using known MACs

This attack uses the fact that for a collision to occur, every intermediate state of $f9$, for the two messages, before the last Kasumi encryption must be equal. Knowing this it is possible to find the K key, using an exhaustive search, independently of K'. Later one also does an exhaustive search to find K'. The attack is split up into three stages:

Find the known MACs Find $2^{(n+m)/2} = 2^{48}$ known MACs and split them into $2^m = 2^{32}$ classes where each member has the same MAC value.

Exhaustive search to find K Take the $\lceil k/m \rceil = 4$, where k is the key length, largest classes. Then compute the input to the last Kasumi encryption in $f9$ for each message and for every key in the current class. If the derived input yields a collision with the original MAC of the current class then increase the key's count with one. It is highly likely that the key with the highest count is the correct one.

Exhaustive search to find K' Now that we know the value of K, an exhaustive search for K' can be conducted. However, in the context of 3GPP we know that the xor difference between K and K' is the KM (Key Modifier). K' can therefore easily be extracted.

Complexity of the attack The attack needs $2^{k+(n-m)/2} * q = q * 2^{80}$ encryptions and $2^{(n+m)/2}$ known MACs, where q is the number of blocks of size n of the plaintext.

This is of course out of bounds in the 3GPP environment.

8.4.4.4 A key recovery attack using MAC verifications

For this attack a large number of verifications and only a few known MACs are required.

The attacker knows a $MAC(X) = M$ for a message $X = X_0, ..., X_1$. With this knowledge he/she can add a block A, with length of one block, to the beginning of X such that $X' = A|X$, and verify if the $MAC(A|X)$ equals M. If this is true then we know that we either have found a collision or that $Kasumi_K(A) = 0$. This operation has a cost of 2^{64} verifications.

If only one A gives the desired result, we can proceed, or else we have to check all our candidate As with another known MAC.

When we have only one block A left we can carry on with the attack. We know now that $Kasumi_K(A) = 0$, and we may find K and K' with exhaustive search.

As you may have understood these attacks required a great deal of verifications and encryptions where the number of encryptions of Kasumi is 2^{129}[37]

8.4.5 Summary

We have seen several types of attacks against the $f9$ integrity scheme, but none which are implementable in the context of 3GPP and Kasumi. However, there may still be unknown attacks that are implementable and may break the $f9$ scheme, but until such an attack is shown we regard the $f9$ as secure enough for the 3GPP environment.

8.5 Conclusion

This chapter has first of all proved that the Kasumi block cipher is provable secure against linear and differential attacks by using the theory of Nyberg and Knudsen. This was done by showing that the complexity of any such attack is beyond reach for today's computers. In addition none of the published attacks (until today August 2, 2004) is a able to break Kasumi, not even produce a small threat.

We have also seen that Kasumi is undistinguishable from a random function, and thus we can label Kasumi pseudo-random. Any block cipher that is pseudo-random has taken a step towards a higher level of security. Thus it is fair to conclude that Kasumi is a strong and secure cipher that will perform well and as intended within the UMTS environment.

Chapter 9

Rijndael - AES

The block cipher Rijndael (AES)[38] was created by Joan Daemen and Vincent Rijmen and submitted as a candidate for the Advanced Encryption Standard (AES) in 1998. Rijndael was made AES in November 2001[38].

This description and review of AES is on a high level, without many details of the construction of AES, only those vital for the use in Milenage (see chapter 10). For a full description of AES please see Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, FIPS-197[38].

9.1 Description

AES is an iterated block cipher build on a Feistel network. The AES cipher takes two inputs; a key K and a block of plaintext P. The output from AES is the ciphertext C. P and C are 128 bits, while K can be either 128, 192 or 256 bits. Depending on the size of the key AES has respectively 10, 12, 14 rounds in it's Feistel network[38].

$$C = AES(P, K) \tag{9.1}$$

In the 3GPP environment AES is only used in encryption mode.

9.2 Properties

Rijndael was not yet made AES when the choice of using Rijndael in Milenage was taken. However, Rijndael fulfilled many of the properties/requirements

needed for the Milenage kernel algorithm. The arguments for choosing Rijndael as the kernel in Milenage are listed in [39].

First of all AES was fast both in hardware and software, and it was suitable for smart card implementations with an 8-bit processor[40, 39]. It uses a small amount of code and takes a small amount of cycles to execute[38]. AES has also a very low memory requirement and a short set-up-time[3]. On a *Motorola 68HC08* 8-bit processor with key length 128, AES uses 8390 cycles, 36 bytes of RAM and a code length of 919 bytes[38].

In addition, it could be protected against side channel attacks without significant speed loss, and it has the required input/output interface. I.e. 128 bit input and output.

It is also relative easy to understand how it functions, making cryptanalysis easier and as a contestant for the AES it has been well studied for some time, without the analysts finding any weaknesses. It was also freely available without cost, do to the enrollment in the competition for the AES title.

9.3 Conclusion

[39] and [41] list several attacks on Rijndael, but no known attack is able to break AES. This is supported by [3] which contains several references to resent attempts to break AES without any luck.

It is therefore safe to consider AES as secure, and can be used within the 3GPP environment. In addition, it is fast and requires only small resources on a smart card (8-bit processor), and can thus be declared close to ideal for UMTS.

Chapter 10

Milenage

In section 6.3.2.1 we discussed the properties of the authentication and key agreement framework. The vital parts of the AKA framework are the eight cryptographic functions ($f0 \rightarrow f5*$), seated in the USIM and the AuC.

Earlier we did not mention the implementation of these functions, only each functions inputs, outputs and requirements (See section 6.3.2.2). This is due to the fact that the 3GPP organization left the implementation, of the functions, to the UMTS operators. However 3GPP did provide an example implementation, for those operators which didn't have the money, time and or capability to implement one for them self.

The presentation of this chapter is mainly based on [39].

10.1 Design criteria

For the general 3GPP requirements to the AKA functions and the cryptographic kernel, please see section 6.3.2.2.

The designers of Milenage added some criteria to the requirements already specified by the 3GPP, and these were:

- The functions shall be indistinguishable from a random source.
- It should be impossible to derive any information, from the output of any function, about the inputs (key, OP)

10.2 Milenage - the example implementation

With the use of the example implementation Milenage the UMTS operators know that it is without major flaws, and that it has undergone thorough testing. Without the proper knowledge and/or resources it may prove difficult to create the AKA functions from scratch, without making any mistakes which can lessen the security.

10.2.1 Meeting the requirements

10.2.1.1 Choice of kernel algorithm

The decision for kernel algorithm landed on, at the time, one of the five finalists for the Advanced Encryption Standard (AES), namely Rijndael. Rijndael is a fast and strong block cipher that had undergone much analysis during the AES contest. In addition Rijndael was highly suitable for smart card implementation, it has good resistance against side channel attacks and it was free of charge. See Chapter 9 for more details about AES.

10.2.1.2 Operator Variant Algorithm configuration Field (OP)

The designers chose to add a 128 bit constant OP to the Milenage algorithm set. The value of OP is for each operator to decide.

The main reason for adding this constant is to create operator specific functions even when using the example functions. This makes the functions different from the other operators, and thus preventing that one USIM can be used on several UMTS networks. In addition, by keeping the value of OP secret, one gains some additional security.

There are two alternatives for storing the value of OP in the USIM. The first alternative is to store the actual value of OP and for each processing derive the value $OP_C = OP \oplus E[OP]$. Or one can store the preprocessed value of OP_C. Then the value of OP is still hidden even if the USIM is compromised.

10.2.2 Functional description

The Milenage algorithm set takes four inputs (K, $RAND$, SQN, AMF) (see section 6.3.2.1) and creates seven outputs. Only f1 and f1* uses all four inputs, while the rest only uses K and $RAND$. The OP is stored on the

USIM as OP_C where $OP_C = OP \oplus E[OP]_K$. The OP_C is then xored with $RAND$ and used as input to the kernel function E.

$$TEMP = E[RAND \oplus OP_C]_K \tag{10.1}$$

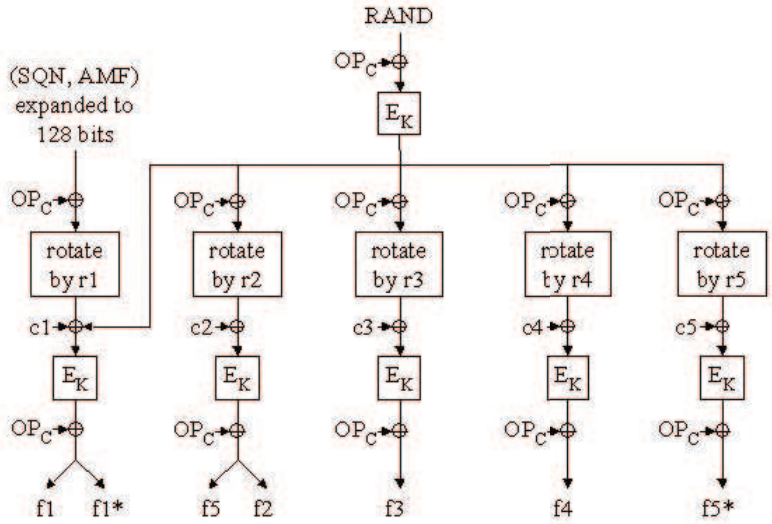

Figure 10.1: Milenage

10.2.2.1 $f2$, $f3$, $f4$, $f5$ and $f5*$

These functions then xors the $TEMP$, from equation 10.1, variable with OP_C before rotating the result ($ROT = TEMP \oplus OP_C$) a different number of times to the left ($r2 = 0, r3 = 32, r4 = 64, r5 = 96$). See figure 10.1. The reason for this is to vary the input, making the output of the functions different. In addition, to this rotation, each rotation result is xored with different constants ($c2$, $c3$, $c4$, $c5$) to once more apply some variations. The product of $ROT \oplus cX$ is then sent through the kernel algorithm $E[ROT \oplus cX] \oplus OP_C$, and then finally xored once more with OP_C.

Each output is xored three times with OP_C and ran two times through the kernel algorithm.

10.2.2.2 $f1$ and $f1*$

The derivation of $f1$ and $f1*$ differs from the rest of the functions in some aspects. Two new inputs to the functions are used, SQN and AMF.

$$ROT = SQN||AMF||SQN||AMF \oplus OP_C \tag{10.2}$$

Equation 10.2 creates the input to the rotate function where ROT is rotated 64 bits to the left ($r1 = 64$). The output from the rotate function is then xored with TEMP (equation 10.1 and the $c1$ constant. The product is then sent in to the kernel function.

10.2.2.3 $f0$

The random function f0 is not included in the Milenage specification, due to the fact that the design and implementation is entirely up to each operator.

10.3 Security analysis of Milenage

When analyzing the Mileage algorithm set it was important to verify two main criteria[39]; The first was to make sure that it was infeasible to retrieve any information about the secret key, or that it is possible to forge output for a given key for a substantial number of $RAND$ values. Secondly to verify that it is infeasible to distinguish the functions from each other.

It should also be taken into consideration that the only probable way that an adversary may gain access to the algorithms is through the USIM. The USIM applies restrictions to what the adversary can control and the speed of the input/output to the algorithms. In the USIM environment the adversary can control the values of $RAND$, SQN and AMF. But only approximately 10 input- output pairs can be generated per second. To mount an attack which requires 2^{64} pairs will be infeasible.

10.3.1 $f1$ and $f1*$

The design of $f1$ and $f1*$ resembles the design of the CBC-MAC operation mode (See section 4.5 for more details about CBC-MAC.), and can be proven equivalent[39]. The CBC-MAC is proven to be secure in the context of 3GPP, as the best possible attack, an internal collision attack (See section 10.3.1.1), requires 2^{64} plain- and ciphertext pairs.

The value of $f1$ is never sent as output from the USIM, and therefore the mentioned attack becomes even more infeasible.

Further, it was proven by the authors of [42] that the CBC-MAC scheme is secure under the notion of the Adaptive distinguisher model (See section 3.3.1) if the underlying kernel function is a pseudo-random function(PRF)[42] (See section 3 for more information.), i.e. cipher block chaining a PRF gives a PRF. So with AES as kernel function the $f1$ and $f1*$ are considered secure.

10.3.1.1 An internal collision attack on $f1/f1*$

The internal collision attack is possible due to the fact that the constructions of $f1$ and $f1*$ are based on the CBC-MAC, and the attack is possible to mount on "any" CBC-MAC construction.

Step one of the attack is to collect 2^{64} input pairs of $x_i = RAND$ and $t_i = SQN||AMF||SQN||AMF$, where x and t is distinct within each pair. Then collect the output z_i.

With a large probability there will exist a collision, i.e. two equal $z_i = z_j, i \neq j$ with different input pairs ($x_i \neq x_j, t_i \neq t_j$). Find this collision, and then it is possible to forge (xor with any 128 bit Δt) new outputs with the help from equation 10.3.

$$f1(x_i, t_i \oplus \Delta t) = f1(x_j, t_j \oplus \Delta t) \tag{10.3}$$

Complexity Due to that the attack has a complexity of 2^{64} derivations of $f1$, pairs (x_i, t_i) and comparisons, it becomes infeasible to mount within the 3GPP environment.

10.3.2 $f2$, $f3$, $f4$, $f5$ and $f5*$

As with the $f1$ and $f1*$ functions, the $f2 \rightarrow f5*$ functions are proven secure under the notion of Adaptive distinguisher model. [39] proves that the advantage the distinguisher has is negligible. The proof thereby gives some assurance that no attack with significantly less calculations than 2^{64} exists.

10.3.3 Separation between $f1$, $f1*$ and $f2 \rightarrow f5*$

[39] has investigated the difference in output between the $f1$,$f1*$ and $f2 \rightarrow f5*$ functions. Their conclusion was that the construction of the functions

and the introduction of the constants ($c2 \rightarrow c5$), in the $f2 \rightarrow f5*$ functions, provides enough separation.

10.3.3.1 Forgery attack against combinations of several modes

Several variations of these attacks exists, but here I will only take one of them as an example. This attack is possible on any of the $f2 \rightarrow f5*$. All different forgery attacks in [39] have a complexity of approximately 2^{64}.

Collect 2^{64} different inputs x_i, and the corresponding outputs of any two functions fI, fJ from $f2 \rightarrow f5*$. Find a collision among the output from different functions.

$$fI(x_i) = fJ(x_j), i \neq j \tag{10.4}$$

Then, accordingly to [39] it is possible to invert the inputs $fI(x_j) = fJ(x_i)$, and thereby obtaining the possibility for forging the output from fI or fJ. This forgery would be highly unlikely if the two functions were independent permutations.

However, these forgery attacks are to be considered infeasible in the 3GPP environment and is no stronger than anticipated[39]

Part III

Conclusion

Chapter 11

Thesis conclusion

The specified security mechanisms of UMTS are the $f8$ and $f9$ functions. The $f8$ function shall provide confidentiality while the $f9$ function provides integrity. The design of them both is built upon known operation modes with a solid background. However, some small modifications where made, and in addition they use a relative new kernel algorithm, Kasumi. Kasumi is a Feistel block cipher, build satisfying the requirements for the provable security against linear and differential attacks. Kasumi is also a pseudo-random permutation (PRP), adding to the trust we have in it for delivering the required security. No attack has either been found that may threaten the confidentiality or integrity of UMTS.

The authentication and key agreement functions can be individually specified by each UMTS provider, however the 3GPP has created an example set; Milenage. Milenage consists of seven different functions required to perform the authentication and key agreement. They all use the AES block cipher as the kernel function. No attack has been found that is able to compromise the Milenage functions, the AKA protocol or the AES block cipher to this date August 2, 2004.

Nevertheless, the security does not rely on these functions alone ($f8$, $f9$, Milenage, AES and Kasumi). They are required, but the security is easily compromised if the keys are not hidden from the adversary. The confidentiality and integrity keys are renewed each time the Authentication and Key Agreement is initiated, but the long term user key is not. All keys are stored on the USIM, and the USIM along with the mobile equipment are easily stolen or simply lost or forgotten on the train etc. It is therefore vital that the USIM (stored on the UICC) is kept secret even to an adversary with physical access to the USIM.

This thesis has examined the different security mechanisms of UMTS, using

block ciphers, to see if they could provide their tasks securely, during the next 20 years, without being compromised. To this question there are no strict *yes* or *no* answer, it all depends. It depends on our trust in the security the mechanisms provide. Do we trust that the users long term key is "impossible" to retrieve from the USIM? Do we trust that there is no attack able to break the $f8$, $f9$, Milenage, Kasumi or AES, existing today or to be found/created within the next 20 years? Is the provable security of Kasumi really provable? $f8$ and $f9$ was "proved" secure under the assumption that the kernel used was a PRP, but this was later proved to be false. Similar "accidents" may happen again.

The description and security assessment of Kasumi has also been given much space in this thesis. As already mentioned the Kasumi cipher is provable secure against linear and differential cryptanalysis. But this gives no protection against other types of attacks. However, it is here the pseudo-randomness steps in. The fact that Kasumi is a family of PRPs tells us that is behaves as a random function, and thus is somewhat protected against known and unknown attacks[1]. Kasumi fulfills all the security properties/requirements we have for block ciphers today , and it is thus safe to consider Kasumi secure, if treated right.

However, should one or more of these functions be compromises, the 3GPP has made sure that almost all functions are replaceable. The Milenage algorithm set are replaceable by definition, the AES as well. The cost will be much higher to replace $f8$, $f9$ and Kasumi, but nothing is impossible. On the other hand, no other block ciphers or operation modes published today can provide significant better security than the ones used by 3GPP. And thus the UMTS providers and users have gotten the *best* security modern cryptography can provide today for the next 20 years.

[1] Remember that perfect secrecy comes from a cipher which has the property that there exist no statistical relations between the plaintext and the ciphertext.

Chapter 12

Further work

This chapter contains some possibilities for further work that can be done within the problem area of this thesis. The work can either be done as a Master thesis or in a Ph. D, in combination with parts of this thesis or with each other.

12.1 A more thorough review of AES/Rijndael

From the time when the cipher Rijndael entered the competition for becoming the Advanced Encryption Standard (AES) it has undergone a great amount of cryptanalysis. The complete overview of different attacks and a full description of the properties, functionality and implementation aspects is a Cand Scient thesis all by itself. The complete review was out of scope for this thesis, but Rijndael is worth a closer look.

12.2 Finding the more exact complexity for a differential attack on Kasumi

As mentioned in section 8.1.5.1 Differential complexity the exact differential complexity of an attack against Kasumi requires a significant more thorough study than I had time to conduct during this thesis.

12.3 A more thorough review of the proof of Kasumi's pseudo-randomness

The proof given in [5] for Kasumi being a pseudo-random permutation is quite complex and inaccessible for the regular cryptographer. Some time should have been spent to give a more educational description. However, once again out of scope and time for this thesis.

12.4 Security of USIM/UICC

The security of UMTS does not rely solely on the $f8$, $f9$ and AKA functions. The long term user key is stored on the USIM and a more thorough assessment of how the key etc. are protected on the USIM is vital for the whole security picture.

Part IV

Appendix

S-boxes

The Decimal tables are retrieved from [25].

S7 - Decimal Table

54, 50, 62, 56, 22, 34, 94, 96, 38, 6, 63, 93, 2, 18, 123, 33, 55, 113, 39, 114, 21, 67, 65, 12, 47, 73, 46, 27, 25, 111, 124, 81, 53, 9, 121, 79, 52, 60, 58, 48, 101, 127, 40, 120, 104, 70, 71, 43, 20, 122, 72, 61, 23, 109, 13, 100, 77, 1, 16, 7, 82, 10, 105, 98, 117, 116, 76, 11, 89, 106, 0, 125, 118, 99, 86, 69, 30, 57, 126, 87, 112, 51, 17, 5, 95, 14, 90, 84, 91, 8, 35, 103, 32, 97, 28, 66, 102, 31, 26, 45, 75, 4, 85, 92, 37, 74, 80, 49, 68, 29, 115, 44, 64, 107, 108, 24, 110, 83, 36, 78, 42, 19, 15, 41, 88, 119, 59, 3

Example: $S7[1] = 50$

S9 - Decimal Table

167, 239, 161, 379, 391, 334, 9, 338, 38, 226, 48, 358, 452, 385, 90, 397, 183, 253, 147, 331, 415, 340, 51, 362, 306, 500, 262, 82, 216, 159, 356, 177, 175, 241, 489, 37, 206, 17, 0, 333, 44, 254, 378, 58, 143, 220, 81, 400, 95, 3, 315, 245, 54, 235, 218, 405, 472, 264, 172, 494, 371, 290, 399, 76, 165, 197, 395, 121, 257, 480, 423, 212, 240, 28, 462, 176, 406, 507, 288, 223, 501, 407, 249, 265, 89, 186, 221, 428, 164, 74, 440, 196, 458, 421, 350, 163, 232, 158, 134, 354, 13, 250, 491, 142, 191, 69, 193, 425, 152, 227, 366, 135, 344, 300, 276, 242, 437, 320, 113, 278, 11, 243, 87, 317, 36, 93, 496, 27, 487, 446, 482, 41, 68, 156, 457, 131, 326, 403, 339, 20, 39, 115, 442, 124, 475, 384, 508, 53, 112, 170, 479, 151, 126, 169, 73, 268, 279, 321, 168, 364, 363, 292, 46, 499, 393, 327, 324, 24, 456, 267, 157, 460, 488, 426, 309, 229, 439, 506, 208, 271, 349, 401, 434, 236, 16, 209, 359, 52, 56, 120, 199, 277, 465, 416, 252, 287, 246, 6, 83, 305, 420, 345, 153, 502, 65, 61, 244, 282, 173, 222, 418, 67, 386,

368, 261, 101, 476, 291, 195, 430, 49, 79, 166, 330, 280, 383, 373, 128, 382, 408, 155, 495, 367, 388, 274, 107, 459, 417, 62, 454, 132, 225, 203, 316, 234, 14, 301, 91, 503, 286, 424, 211, 347, 307, 140, 374, 35, 103, 125, 427, 19, 214, 453, 146, 498, 314, 444, 230, 256, 329, 198, 285, 50, 116, 78, 410, 10, 205, 510, 171, 231, 45, 139, 467, 29, 86, 505, 32, 72, 26, 342, 150, 313, 490, 431, 238, 411, 325, 149, 473, 40, 119, 174, 355, 185, 233, 389, 71, 448, 273, 372, 55, 110, 178, 322, 12, 469, 392, 369, 190, 1, 109, 375, 137, 181, 88, 75, 308, 260, 484, 98, 272, 370, 275, 412, 111, 336, 318, 4, 504, 492, 259, 304, 77, 337, 435, 21, 357, 303, 332, 483, 18, 47, 85, 25, 497, 474, 289, 100, 269, 296, 478, 270, 106, 31, 104, 433, 84, 414, 486, 394, 96, 99, 154, 511, 148, 413, 361, 409, 255, 162, 215, 302, 201, 266, 351, 343, 144, 441, 365, 108, 298, 251, 34, 182, 509, 138, 210, 335, 133, 311, 352, 328, 141, 396, 346, 123, 319, 450, 281, 429, 228, 443, 481, 92, 404, 485, 422, 248, 297, 23, 213, 130, 466, 22, 217, 283, 70, 294, 360, 419, 127, 312, 377, 7, 468, 194, 2, 117, 295, 463, 258, 224, 447, 247, 187, 80, 398, 284, 353, 105, 390, 299, 471, 470, 184, 57, 200, 348, 63, 204, 188, 33, 451, 97, 30, 310, 219, 94, 160, 129, 493, 64, 179, 263, 102, 189, 207, 114, 402, 438, 477, 387, 122, 192, 42, 381, 5, 145, 118, 180, 449, 293, 323, 136, 380, 43, 66, 60, 455, 341, 445, 202, 432, 8, 237, 15, 376, 436, 464, 59, 461

Abbreviations

In order of appearance.

3G - Third Generation

UMTS - Universal Mobile Telecommunications System

AES - Advanced Encryption Standard

STM - Symmetric-key trust model

MAC - Message Authentication Code/Scheme

A - Alice/Sender

B - Bob/Reciever

O - Oscar/Opponent/Adversary

SES - Symmetric-key encryption Scheme

DES - Data Encryption Standard

FN - Feistel network

PRF - Pseudo-Random Functions

PRP - Pseudo-Random Permutation

SRF - Shared-Random Function model

CTR - Counter mode (encryption scheme)

ECB - Electronic Code Book

CBC - Cipher Block Chaining

OFB - Output Feedback mode

CBC-MAC - Cipher Block Chaining - Message Authentication Code

SPA - Simple Power Analysis

DPA - Differential Power Analysis

1G - First Generation

2G - Second Generation

3GPP - Third Generation Partnership Project

GSM - Global System for Mobile communication (2G)

UTRA - Universal Terrestrial Radio Access

ETSI - European Telecommunications Standards

IMT-2000 - International Mobile Technology

UE - User Equipment

ME - Mobile Equipment

USIM - UMTS Secure Identity Module

UICC - Universal Integrated Circuit Card

UTRAN - UTRA Network

AN - Access Network (short for UTRAN)

Node-B - UMTS Base station

BS - Base Station

RNC - Radio Network Controller

CN - Core network

AuC - Authentication Center

HE - Registers Home Environment

EIR - Equipment Identity Register

HLR - Home Location Register

CS - Circuit Switched

PS - Packet Switched

SGSN - Serving GPRS Support Node

GPRS - General Packet Radio Service

MSC - Mobile Services Switching Centre

VLR - Visited Location Register

IMSI - International Mobile Subscriber Identity

AKA - Authentication and Key Agreement

August 2, 2004

IK - Integrity Key

CK - Confidentiality Key

AK - Anonymity Key

AV - Authentication Vector

MS - Mobile Station (ME + USIM)

SN - Serving Network

OP - Operator Variant Algorithm Configuration Field

AMF - Authentication Management Field

TMSI - Temporary Mobile Subscriber Identity

LAI - Location Area Identification

KSI - Key Set Identifier

RKA - xor-restricted Related-Key Attack

LOR - Left-Or-Right indistinguishability

FIPS - Federal Information Processing Standard

Bibliography

[1] A. Menezesa and P van Oorschot and S. Vanstone, *Handbook of Applied Cryptography* (CRC Press, 1996).

[2] P. R. Mihir Bellare, *Introduction to Modern Cryptography* .

[3] A. Biryukov, *Block Ciphers and Stream Ciphers: The State of the Art* (IACR ePrint, 2004).

[4] C. E. Shannon, *Communication Theory of Secrecy Systems* (, 1949).

[5] O. Yi, J.-S. Kang, S.-U. Shin, and D. Hong, *Provalbe Security of KASUMI and Encryption Mode f8* .

[6] W. Stallings, *Network security essentials: Application and standards* (Prentice Hall, 2000).

[7] D. R. Stinson, *Cryptography - Theory and Practice* (Chapman & Hall/CRC, 2002).

[8] E. Biham and A. Shamir, *Differential Cryptanalysis of DES-like Cryptosystems* (, 1990).

[9] X. Lai and J. L. Massey, *Markov Ciphers and Differential Cryptanalysis* (, 1991).

[10] E. Biham and A. Shamir, *Differential Cryptanalysis of DES-Like Cryptosystems* (, 1990).

[11] E. Biham, *Differential Cryptanalysis* (, 2000).

[12] M. Matsui, *Linear cryptanalysis method for DES cipher* (, 1993).

[13] . G. P. Project, About 3gpp.

[14] Michael Walker, *On the Security of 3GPP Networks* (, 2000).

[15] 3rd Generation Partnership Project, *TS 33.102 Security architecture*, v5.2.0 ed. .

[16] 3rd Generation Partnership Project, *3GPP TR 21.905. Technical Specification Group Services and System Aspects. Vocabulary for 3GPP Specifications*, v6.3.0 ed. .

[17] 3rd Generation Partnership Project, *TS 35.206 Specification of the Milenage Algorithm Set; An exaple algorithm set for the 3GPP authentication and key generation functions f1, f1*.f2.f3.f4.f5 and f5*;Document 2: Algorithm Specification*, v5.1.0 ed. (, 2003).

[18] 3rd Generation Partnership Project, *TS 33.105 Cryptographic Algoritm Requirements*, v4.1.0 ed. .

[19] 3rd Generation Partnership Project, *TS 35.205 Specification of the Milenage Algorithm Set; An exaple algorithm set for the 3GPP authentication and key generation functions f1, f1*.f2.f3.f4.f5 and f5*;Document 1: General*, v5.0.0 ed. (, 2002).

[20] 3rd Generation Partnership Project, *TR 33.908 General Report on the Design, Speification and Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms*, v5.2.0 ed. .

[21] 3rd Generation Partnership Project, *Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms*, Sage version 2 ed. .

[22] M. Matsui, *New Block enctyption Algorithm MISTY* .

[23] K. Nyberg and L. R. Knudsen, *Provable Security Against a Differential Attack* (, 1995).

[24] K. Nyberg, *Chapter Three: Cryptographic Algorimtms for UMTS* (, 2004).

[25] 3rd Generation Partnership Project, *TS 35.202 Specification of the 3GPP Confidential and Integrity Algorithm. Document 2: KASUMI specification*, v5.0.0 ed. .

[26] M. Matsui*On a Structure of Block Ciphers with Provable Security against Differential and Linear Cryptanalysis* Vol. E82-A (, 1999).

[27] K. Aoki and K. Ohta, *Stricter Evaluation for the Maximum Average of Differential Probability and the Maximum Average of Linear Probability* (, 1996).

[28] M. Blunden and A. Escott, *Related key attacks on reduced round Kasumi* (, 2002).

[29] T. Iwata and K. Kurosawa, *New Security Proofs for the 3GPP Confidentiality and Integrity Algoritms* (, 2004).

[30] M. Luby and C. Rackoff, *How to construct pseudorandom permutations and pseudorandom functions* (SIAM J. Comput, 1988).

[31] K. Sakurai and Y. Zheng, *On non-pseudorandomness from block ciphers with provable immunity against linear cryptoanalysis* (IEICE Trans. Fundamentals, 1997).

[32] J.-S. Kang, O. Yi, D. Hong, and H. Cho, *Pseudorandomness of MISTY-Type Transformations and the Block Cipher KASUMI* (Springer-Verlag, 2001).

[33] H. Gilbert and M. Minier, *New results on the pseudoransomness of some block cipher constructions* (Preproceedings of Fast Encryption workshop 2001, 2001).

[34] M. Bellare and A. Desai and E. Jokipii and P. Rogaway, *A Concrete Security Treatment of Symmetric Encryption* (, 1997).

[35] K. Kurosawa and T. Iwata, *On the correctness of security proofs for the 3GPP confidentiality and integrity algoritms* (, 2003).

[36] M. Bellare, J. Kilian, and P. Rogaway, *The security of the cipher block chaining message authentication code* .

[37] L. R. Knudsen and C. J. Mitchell, *An analysis of the 3GPP-MAC-scheme* (, 2000).

[38] N. I. of Standards and T. (NIST), *Federal Information Processing Standards Publication 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES)* (NIST, 2001).

[39] 3rd Generation Partnership Project, *TR 33.909 Report on the Design and Evaluation of the MILENAGE Algorithm Set; Deliverable 5: An Example Algorithm for the 3GPP Authentication and Key Generation Functions*, v4.0.1 ed. .

[40] J. Nechvatal *et al.*, *Report on the Development of the Advanced Encryption Standard (AES)* (, 2000).

[41] J. Daemen and V. Rijmen*AES Proposal: Rijndael* Vol. Document version 2 (, 1999).

[42] M. Bellare, J. Kilian, and P. Rogaway, *The security of Cipher Block Chaining* (, 1994).

List of Figures