Linköping Studies in Science and Technology
Dissertation No. 722

Codes for Digital Fingerprinting

Jacob Löfvenberg

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden

Linköping 2001

ISBN 91-7373-130-7
ISSN 0345-7524

Printed in Sweden by UniTryck, Linköping 2001.

To my great pleasure

Abstract

Unlicensed, or illegal, copying of data is a problem in many areas. Despite various efforts in copy protection and copyright enforcement (both legal and technical), the problem still exists, and with the growing use of digital means of storing and distributing data the problem seems to be getting worse.

Storing a unique, invisible marking in each distributed object is a possible way of dealing with the problem of illegal copying. That way, if an illegal copy is found somewhere, it is possible to find the original holder of the object and to take legal action. This type of copyright enforcement scheme is called fingerprinting.

This thesis deals with codes for fingerprinting of digital data in the context of several users colluding to create an untraceable, illegal copy (such users are called pirates). Some theoretical bounds on the performance of such codes are presented. A general method to statistically test whether certain users have taken part in the creation of a particular illegal copy is presented, and pirate strategies that prevent this kind of testing are derived. A simple testing method is presented that tests whether a certain, proposed group of users should be considered guilty of having created a specific illegal fingerprint. The method's performance is analysed, and it is found that the error probability is practical for an interesting range of values of the number of colluding pirates and the length of the fingerprints. The set of possible pirate strategies is derived and a compact representation is presented. It is shown that the fingerprinting problem is a general game theoretical problem. Finally, a combinatorial performance measure for binary fingerprinting codes is presented, and several code constructions are analysed using this measure.

Acknowledgments

Writing this thesis is something I have not done entirely on my own. Many people have been there to help and support me, and I would like to take this opportunity to sincerely thank some of them:

Dr. Niclas Wiberg was my supervisor while I was writing my licentiate thesis (approximately chapters 2 to 5 in this thesis). He was a never-ending source of new ideas and inspiration.

Professor Ingemar Ingemarsson, who has been my supervisor for the rest of my doctoral studies. He has always shared his enthusiasm, reading everything and pointing out places where the text has been hard to follow or where further discussion has been needed.

All of my colleagues at Information Theory, Image Coding and Data Transmission. A special thanks to Tina Lindkvist, with whom I have been sharing both office and research interest for more than five years. I have had a great time.

My parents, for their love and support, and for being so interested in my work all these years I have spent at the university.

Malin, for her love and support, and for just being there.

I would also like to thank Teknikvetenskapliga Forskningsrådet (Swedish Research Council for Engineering Sciences) for the financial support for my research.

Thank you, all of you.

Linköping, June 2001

Jacob Löfvenberg

Contents

Chapter 1

Introduction

This work deals with the problem of handling illegal copying of digital data. More specifically, the use of individual, user-unique markings of the data, fingerprints, is examined for the purpose of tracking the origin of a discovered, illegal copy of the data.

The scenario is the following. Somebody, the distributor, has a digital object that he distributes to a number of users after having individually and uniquely marked each user's copy with a fingerprint. A number of users, called pirates, cooperate in creating an illegal copy that they redistribute. The distributor finds one of the illegal copies and tries to find at least one of the users who took part in the creation of the redistributed copy.

The area of fingerprinting is a rather new one and there are no standard questions to ask or standard models to use. The choice of topics in our research has been made based on what have seemed to be promising problems to work with at each given time, something that might explain the disparity of the different chapters. We choose to consider this, not as a weakness, but as a strength – there are many ways to view the fingerprinting problem and thus many ways to describe and shed light on it, many of which give rise to interesting research problems. This is of course especially nice when writing a thesis on the subject.

Questions that occur naturally in fingerprinting research include the following:

- What does the fingerprinting system look like? Who are the different actors, the different actions, the different objects, and how does everything fit together?

- What is performance and how do we measure it? Interesting parameters in a fingerprinting system are, for example, number of users, number of pirates working together in creating the illegal copy, length of the fingerprints and some performance measure. Usually the performance measure is chosen either as some kind of combinatorial measure stemming from the fingerprints used, or as the probability of failing to find the perpetrators.

- How do we achieve good performance? This question breaks down into two different ones: how to choose the fingerprints and what method to employ when trying to find the pirates. Both of these questions are dependent on how the pirates behave when creating the illegal copy. This in turn results in the question of what the pirates can do to prevent the distributor from finding them.

These questions are not fully answered in this work, but they form a framework in which the different chapters of this thesis fit. Some especially interesting, open areas are pointed out in chapter 9.

This dissertation is an extension of the author's licentiate thesis *Random Codes for Digital Fingerprinting*, [JL99], where the contents of chapters 3 to 5 are very similar to chapters 3 to 5 in this work. The contents in the new chapters of this work leave the restriction of using random fingerprinting codes and discuss some new topics in a more general setting.

1.1 Outline of the Thesis

Chapter 2 begins with a rather thorough discussion about fingerprinting. This is followed by a description of related research areas and a description of the model and the notation we are using.

Chapter 3 describes two different theoretical bounds on the performance of fingerprinting systems. One is a general bound on the fingerprint length needed for it to be possible to find collusions of pirates and one is a bound on the probability of not succeeding in tracing a pirate when the pirates use a very simple strategy. The latter bound is dependent on the code used and it is specifically applied to systems based on the kind of random code that is used in chapters 4 and 5.

Chapter 4 discusses correlation-based testing methods, and a way is shown for pirates to prevent the use of correlation-based testing methods for finding subsets of the pirate group. This entire chapter assumes the use of a certain kind of random code.

Chapter 5 contains a description of a specific method to test whether or not a proposed group of users should be considered guilty of having created a certain illegal copy. This chapter relies on [JL98], a paper by the author, presented at the 1998 International Symposium on Information Theory, MIT, Cambridge, USA. The same code as in chapter 4 is used.

Chapter 6 examines, in a very general setting, the pirates' freedom of action when creating the illegal copy. The result is a compact representation, able to describe any pirate strategy.

Chapter 7 introduces basic concepts from game theory and shows that fingerprinting is a game in the game theoretical sense. This is a fundamental property of fingerprinting and implies that analysis and constructions of fingerprinting systems in general have to be game theoretical. The game is a general matrix game and not some special case that is easier to analyse.

Chapter 8 defines a combinatorial performance measure for fingerprinting codes. Several code classes are defined and their performance in the sense of the defined measure is analysed in the case of two pirates.

Chapter 9 proposes directions for future research in the area of digital fingerprinting.

Chapter 2

Background

2.1 Problem Background

2.1.1 Introduction

Illegal copying is a major problem in many areas. This is especially true when the products copied consist mainly of data (for example, movies, computer software and audio CD:s). Different “data products” are copied for different purposes all over the world. What is common is that the holders of the rights to the products almost always lose money because of the copying.

Traditionally illegal copying has been kept at a tolerable level by a combination of law enforcement, distribution problems and the fact that copies usually have a lower quality than the original. With the enormous growth of Internet technology and digitally stored data, this is no longer so. It is now easy and cheap to make identical copies of an original and to make the copies available to the entire Internet community.

Our proposition is to accept that while illegal copying is impossible to prevent, it can be made less attractive by making the risk of being caught greater. The way we propose to increase this risk is by using fingerprinting.

The core idea of fingerprinting is that each user receives a copy of the object in question, containing a unique marking. The marking can be used to identify the object, and thereby also the user if his identity is linked to the fingerprint in some way, for example by distributing copies only to users who identify themselves.

2.1.2 Why the Problem is Worse now than Before

Earlier illegal copying and distribution have been kept at a tolerable level by a combination of different factors. Such factors are laws, distribution problems and quality factors.

Law enforcement has ensured that large scale copying with expensive, high quality equipment and high quality copies has been impossible to get away with, as it has demanded very much in terms of employees, retailing channels and investments in equipment. It has not been possible to keep such a large scale business secret for long enough to make it worth the investment.

Small scale copying implies cheaper equipment and thereby also lower quality of the equipment and the copies. The quality degradation has made the copies less attractive to use, and thereby the usage of low quality copies has been limited.

The problem of distribution has also been a deterrent. Large scale distribution from a central source has been difficult since it has not been possible to openly advertise and deal with illegal copies without being noticed by authorities. Distribution without a central source has been ineffective and difficult. For example, making copies only between people who know each other makes the propagation of new material slow and unreliable.

The combination of the Internet and digitally stored data has done away with these restrictions, at least for products containing only digital data. Everybody can make identical copies of digital data using cheap technology, and “identical” means that there is no quality degradation at all. Using the Internet it is easy to make the copies available for everybody in the Internet community, and because of the large number of Internet users, this can only be considered very large scale distribution. Because of both technical and legal

issues it is also difficult to find and prosecute the person guilty of the illegal copying in such cases.

In summary we can say that because of data being digitally stored, and the widespread use of the Internet, it is now possible to do cheap, large scale, high quality copying with a low risk of being caught. This of course opens entirely new possibilities for people interested in making or using illegal copies. A recent example is Napster – the Internet service that allowed anybody with Internet access to download audio files encoded with MPEG-1 Audio Layer 3 (mp3-files).

2.1.3 Why Copy Protection does not Work

Different approaches to prevent illegal copying have been tried. Some computer programs cannot be installed unless a certain serial number, or code, is entered, and hardware solutions have been used for preventing illegal copying of DAT tapes and DVD disks. The fundamental problem with this is that the data *has* to be accessible to the users when they want it, and at that moment there is no way to guarantee that users do not make a copy of the data in a way that they control. In this way they can remove the protection and then do with the information what they want. This might not be an easy thing to do in practice, but it is always possible in principle. Further, if the information is initially stored in a general purpose computer at the user's, which is quite likely if they have got the information from the Internet, removing the copy protection is just a matter of programming the computer.

Advanced methods using strong encryption do not solve the problem either. As mentioned above, the users have to be able to use the data as they like, and to be able to use the data it must be decrypted. At that point the data is unprotected and it is possible for users to copy it, in its unprotected form, to some other place that they control. By doing so they have removed the protection from this piece of data.

The analysis so far of the current situation is the author's own, but it seems to be shared with everyone else seriously discussing the subject. Still, it is a subjective analysis, and has no research basis. A good source of discussion about

these things, and information security issues in general, is Bruce Schneier's free monthly e-mail newsletter Crypto-Gram [BSCG].

2.2 A Fingerprinting Discussion

2.2.1 The Idea of Fingerprinting

What we propose instead of trying to prevent illegal copying is to try to deter people from it. This can be done in at least two different ways: by making the punishments for those caught harder and by making the risk of being caught greater. The former is a matter of legal policy and not something we will discuss here. The latter, on the other hand, is something that we can affect by technical means, and this is where fingerprinting comes in.

The idea of fingerprinting is to mark each copy uniquely, so that every distributed copy is a little different from every other. In this way it is possible to distinguish between all legal copies. Thus, if we distribute copies only to persons (legal or physical) who identify themselves, it might be possible, if an illegal copy is found, to identify the person who bought the legal copy from which the illegal copy was made.

Doing so, we hope, will deter users from making illegal copies, since if they spread the copy to somebody else, they will lose control over it, and it is possible that the copy will arrive at somebody who will identify and prosecute the user for creating and spreading the copy. Thus the greater risk of being identified ought to make it less popular to spread illegal copies.

The marking, or fingerprint, has to be such that it is not visible to the user, and such that it is difficult to remove, both by mistake and if the user actively tries to remove it. The fingerprint also has to have the property that if the object is copied, the fingerprint will also be copied, so that the new copy contains the same fingerprint as the original.

A very natural, but unfortunately bad idea, is to put the fingerprint somewhere where it does not affect the data. For example, in text documents this could be

in unused formatting codes, or by coding the fingerprint as small alterations in the space between letters or rows. This is bad because by copying only the interesting data it is possible to remove the marking. In the text document case this could be done by just retyping the entire document in a word processor; a tedious task, but a very simple one. The solution, as we see it, is to put the marking *in the data that forms the product*.

That it is actually possible to fingerprint data is not obvious at all, but we have found it to be possible in a large number of cases. We have also found cases where we are convinced that it does not work (for example poetry and medical images). The fingerprinting idea we have is that in some places in an object it will be possible to exchange some small part with something else, without this destroying the object, and without this being easily seen. In every such place we can encode, or store, a single digit by choosing either the original small piece or one of the alternative small pieces. For example, in a text document this could be single words that may be replaced with synonyms.

We will not dwell on how easy or difficult this is to do, but rather assume that it is possible. What is interesting with this way of making fingerprints (encoding digits individually by making alterations in the object) is that we separate the fingerprinting problem into two subproblems that can be addressed individually: the embedding problem and the coding problem.

The embedding problem is the problem of how to make the alterations by which we encode single digits into the object. The coding problem is the problem of how to choose the fingerprints in their abstract representation (as vectors of digits) such that the fingerprinting system becomes robust against different types of attacks. For the embedding problem we think there is a need for a deep understanding of the kind of data we want to use the fingerprinting system on. What kind of, and how much, alteration is acceptable is a very difficult problem. The coding problem, given a suitable model of what is possible to do for the owners of fingerprinted objects, is a purely mathematical problem, and is the major part of what we are going to address in this thesis.

2.2.2 Fingerprinting Goals

If an illegally copied object (illegal copy, for short) is discovered, we can have two different goals:

- to be able to identify the pirate or, in the case of several pirates working together, as many of the pirates as possible.
- to be able to test whether a certain, proposed user, or group of users, is guilty of having created the illegal copy.

In both of these cases we will do this by first recovering the fingerprint of the illegal copy and then using a database containing the users' fingerprints together with some kind of algorithm to try to accomplish our goal.

2.2.3 A Fingerprinting Example

To show the basic ideas of fingerprinting, we will give an example where many of the concepts are illustrated.

Example 2.1: Assume that we have a short text that we want to fingerprint. Our reason might be that somebody inside our organization is leaking information about our next generation products to the Internet, and we want to find out who it is. The original text looks like:

> The fantastic new technology that we in New Technology Inc. have discovered is of profound importance to new generations of cars.

As authors of this text we think that the word substitutions in table 2.1 can be made without the meaning being lost.

Original / ‘0’	Variant / ‘1’
fantastic	amazing
discovered	invented
profound	great
new	future
cars	automobiles

Table 2.1: *Possible word substitutions in example text.*

This list of possible substitutions must, of course, be kept secret from the users. Should the users know which words could be exchanged for which other words, it would be simple for users leaking the information to remove any traces of their fingerprints from the illegal copy they create.

Each of these substitutions can be made independently of the others, which means that we have five places in which we can choose freely between two different words. If we represent the original word with a ‘0’ and the variant with a ‘1’, we can describe any version of the text with a five-bit binary vector. Furthermore, every five-bit binary vector has its corresponding version of the text. This means that there is a one-one correspondence between the set of binary vectors of length five and the set of all possible versions of the text.

The original text corresponds to the vector (00000), the text “The amazing new technology that we in New Technology Inc. have discovered is of profound importance to future generations of cars” corresponds to the vector (10010), and the vector (01001) corresponds to the text “The fantastic new technology that we in New Technology Inc. have invented is of profound importance to new generations of automobiles”.

If we think that our leak works alone, we can distribute this text in $2^5 = 32$ different versions to as many different people. If we remember who receives which version, we can find the leak if we happen to find the text somewhere on the Internet. This is the simplest possible way to use fingerprints. If two people with different copies of the text work together, they can create a new

text that differs from both of their own texts, and thereby frame some other, innocent user. This is the kind of problem that is examined in this thesis.

❑

2.2.4 Equivalence Classes

The fact that the substitution table in example 2.1 has to be kept secret from the users can be generalized. We must demand that the kind of data from which the variants are chosen cannot be partitioned into equivalence classes, or that the equivalence classes are not known to the pirates. In example 2.1, if words in general could be partitioned into equivalence classes (exact and universally valid synonyms), people leaking the information could just choose the first synonym in lexicographical order from the word's equivalence class for every word in the text. This way they would create a new text with exactly the same meaning, which could not possibly be used to identify them.

This means that the data being fingerprinted has to be such that there are "almost synonyms", or "sometimes synonyms". It has to be possible to create objects that are sufficiently similar to the original, and it has to be difficult for others than the author, or creator, of the original to decide what is sufficiently similar. That way it is relatively easy for the copyright holder to create objects that are sufficiently similar, but it is difficult for anybody else to create an illegal copy of their own copy that is both sufficiently similar to the original and so unlike their own copy that the illegal copy cannot be traced back to them.

2.3 Related Areas

The descriptions and references in this sections are in no way complete. The purpose is instead to give starting points for further reading and to show how this thesis fits into a bigger picture. A short but good overview of the relation between the areas described below can be found in [HK99], by Hartung and Kutter. The book [KP00], edited by Katzenbeisser and Petitcolas, is an up-to-date description of the areas steganography, watermarking and fingerprinting.

2.3.1 Steganography

Steganography is about embedding a secret message in a cover message so that it is not visible and so that it cannot be found. The embedding can be parameterized by a key that is used when retrieving the secret message. The secret message may be encrypted for added security, but the encryption cannot do what steganography does, which is hide the existence of the secret message.

An overview of steganography, with several further references, has been made by Anderson and Petitcolas [AP98]. A game theoretical analysis of steganography has been presented by Ettinger [JME98]. This work is the closest we have found to our chapter 7, about game theory in the fingerprinting setting. There are big differences though, since in the fingerprinting case, as opposed to the steganographic setting in [JME98], the opponent (the pirates) know several of the positions where information is hidden, and can use this to make a much more sharply targeted attack.

2.3.2 Watermarking

Watermarking is much like steganography, only that the scheme should be robust against active attackers, even if they know not only that an object contains a watermark, but also if they know the algorithmic principle of the method. The proposed application for watermarking is usually copyright information/validation.

In [HK99], by Hartung and Kutter, there is a thorough presentation of the watermarking field and a large number of references. Several, not so technical, papers on watermarking can be found in a special issue of Communications of the ACM, [ACM98].

2.3.3 Traitor Tracing

The traitor tracing problem is the following: A distributor broadcasts encrypted data that should be available only to a certain set of users. Each user has a unique decryption key that can be used to decrypt the broadcast data. Some users will collude and create a new key, different from any of

theirs, but still able to decrypt the broadcast data. In a traitor tracing scheme, if the number of users colluding is less than some given threshold, it should be possible, with low probability of error, to trace at least one of the creators of the new key.

Traitor tracing was first introduced by Chor, Fiat and Naor in [CFN94], and later extended and updated in [CFNP00]. Some other papers in the area are by Boneh and Franklin [BF99] and Naccache, Shamir and Stern [NSS99].

2.3.4 Fingerprinting

Fingerprinting contains elements from all the above areas. Information is hidden in other information. This hiding has to be robust against attacks and if several users collude and create a new object (corresponding to a new key in the traitor tracing scenario), it should be possible to find at least some of the users who colluded.

The earliest paper on fingerprinting that we have found is by Wagner [NW83], which contains a general description of the idea of making otherwise identical objects unique. The description is not limited only to digital objects.

The first paper on fingerprinting in the presence of collusions was by Blakley, Meadows and Purdy [BMP85], where a specific fingerprinting scheme is presented in which the number of copies an opponent must obtain in order to erase the fingerprints is specified.

An important paper in the area of collusion secure fingerprinting is by Boneh and Shaw [BS95], [BS98]. In the first of these papers the Marking Assumption was stated explicitly for the first time. Chapter 6 in this thesis, about pirate strategies, is really a specification of this assumption – a systematic study of what the Marking Assumption's implies about the pirates' possible actions. The part of [BS98] that is most closely related to this thesis is a construction of probabilistically collusion secure codes. The code is defined and analysed and the error probability is calculated. By concatenating the code with a random outer code it is shown that given a certain error probability the required fingerprint length grows $O(c^4)$ with the number of pirates, c, and

$O(\log M)$ with the number of users in the system, M. A lower bound on the length, n, of fingerprinting codes that can handle c colluding pirates with an error probability ε is also shown: $n \geq (1/2)(c-3)\log(1/\varepsilon c)$.

There are several papers on codes with the identifiable parent property (i.p.p.): Staddon, Stinson and Way [SSW00], Hollman, van Lint, Linnartz and Tolhuizen [HvLLT98], Barg, Cohen, Encheva, Kabatiansky and Zémor [BCEKZ00]. These papers present properties and constructions for i.p.p. codes. In chapter 8 of this thesis this concept is generalized to allow measuring how close to being i.p.p. a code is when it is not an i.p.p. code. In the beginning of that chapter, work related to i.p.p. codes is somewhat more thoroughly described.

Asymmetric fingerprinting is described by Pfitzmann and Schunter [PS96], and anonymous fingerprinting is described by Pfitzmann and Waidner [PW97], and Pfitzmann and Sadeghi [PS99].

2.4 Model

This section introduces our fingerprinting model, its actors, actions and objects. The model is not only a prerequisite for analysing the fingerprinting problem, it can also be used as a framework in which the contents of the different chapters in this thesis can be placed and understood. We will restrict ourselves to work only with binary fingerprinting systems.

We will describe two slightly different models: the random code model and the known code model. The random code model is used in chapters 3 to 5, and the known code model in chapters 6 to 8.

The contents of the known code model in this section were assembled in cooperation with Tina Lindkvist, also a Ph.D. student at the Department of Electrical Engineering, Linköping University.

2.4.1 Objects, Actors and Actions

An *object* is a collection of digital data. Digitally stored texts, images or audio files are examples of such objects.

An object is *close* to another object if they are so similar that they serve the same purpose for normal use. This means that close has different meanings for different people and depends on what the object is used for. However, we will assume that in the fingerprinting system under consideration everybody involved agrees on what is close and what is not.

An *original* is an object that somebody, who has the legal right to do so, wants to distribute. There is exactly one original in the fingerprinting system. For example, a piece of music, a book or an image in digital form can be an original in this sense.

The *distributor* is the sole entity that has the original, and the right to distribute it. For example, if the original is a book the distributor can be the publisher of that book, that is, the person/company which has the right to distribute the book.

A *copy* is an object close to the original. A copy in this meaning is not an exact digital copy digit by digit. Instead it is an object that is so similar to the original that it for all purposes serves just as well.

The distributor distributes copies to the *users*. The number of users is M, and the set of users is denoted $U = \{1, 2, \ldots, M\}$. A user is a rightful owner of a copy. Every user has the distributor's permission to hold and use a copy.

Illegal distribution is the distribution of copies by somebody else than the distributor. The distributor is the only one who is allowed to distribute copies of the original. All other forms of distribution are considered to be unwanted by the distributor, and thus something he or she wants to avoid.

Pirates are users performing illegal distribution. The number of pirates is c, and the set of pirates is denoted $P = \{p_1, p_2, \ldots, p_c\} \subseteq U$. If P contains more than one pirate, it is called a *collusion*. All pirates cooperate.

Fingerprinting is the act of distorting an original in different ways for every user so that each user's copy is unique, while still being close to the original.

The pirates will distribute to a person, other than the users, who wants a copy. This is not allowed, and to discourage this the distributor will use fingerprinting.

2.4.2 The Code in the Known Code Model

A *code* is an M-tuple such that the elements of the tuple together form a subset of size M of an n-dimensional vector space over GF(2).

The *public code* is a code denoted $\Psi = \langle \psi^{(1)}, \psi^{(2)}, \ldots, \psi^{(M)} \rangle$. The elements of the public code are binary vectors $\psi^{(i)}$ of length n, such that $\psi^{(i)} \neq \psi^{(j)}$ when $i \neq j$. The public code is not secret.

Definition 2.1: [1]Let C_1 and C_2 be two codes of the same size and length. We say that C_1 is *equivalent* to C_2 if C_1 can be transformed into C_2 by the following operations:

1. Permutation of the coordinates.
2. Permutation of the codewords.
3. Addition modulo-2 with the same binary vector of length n to every codeword.

The *embedded code*, Γ, is a code equivalent to Ψ. Γ is chosen at random by the distributor, with equal probability for all codes equivalent to Ψ. For interesting code sizes this class of equivalent codes contains a large number of elements, so the probability for any specific code of being chosen as the embedded code is small. Γ is kept a secret by the distributor.

This way of describing the embedded code, as a code randomly chosen among the codes equivalent to the public code, is a way of modelling that the

1. Note that there are other ways of defining code equivalence that are more common than the one used here. The reason for this definition is the code definition above that is also different from what is usually seen.

users know what the fingerprinting code looks like, but that they cannot tell the difference between zeros and ones when they are embedded and that they do not know which user gets which codeword. We can thus think of the embedded code Γ as the result of embedding the publicly known fingerprinting code into the fingerprinted objects.

2.4.3 The Code in the Random Code Model

A *code* is a subset of size at most M of an n-dimensional vector space over GF(2).

In this model the code is constructed randomly, each codeword chosen from a uniform distribution over the set of n-dimensional binary vectors. This means that every bit in every codeword is statistically independent of every other bit in any codeword, and that for every bit the probability of '1' and '0' is equal. Also in the random code model it is impossible for the pirates to tell the difference between embedded zeroes and ones.

Since the codewords are chosen randomly, we do not know what properties the actually chosen code has. The codewords are not even necessarily all different, though for reasonable code lengths it is highly improbable that this would not be the case. The possibility of a bad code being chosen is taken into account when calculating the resulting error probability, so we still have control over this.

2.4.4 Distribution

A *mark* is a bit encoded in an object so that '0' is encoded as no change from the original, and '1' is encoded as some specific change from the original. Every object contains n marks. The positions of the marks are also kept secret by the distributor. A *detectable mark* is a mark that differs between compared objects. An *undetectable mark* is a mark that is the same in all compared objects.

We will sometimes need to add marks, and will do this by identifying the different states with the elements in GF(2). Addition then means addition modulo-2.

Allowing only one alternative to the original in each mark means that we restrict ourselves to work with binary codes. It is possible to use more than one alternative for each mark, thus getting a system that is non-binary.

The following assumptions are made about the marks:

1. No mark is affected by any other mark.
2. An object is close to the original regardless of which combination of '0's and '1's the marks encode.
3. Colluding pirates can find a specific mark if and only if it is detectable. They cannot change an undetectable mark without destroying the fingerprinted object. For any detectable mark the pirates can create a new object in which that mark contains '0', '1' or '*e*', where '*e*', called an erasure, denotes anything but '0' and '1'. Sometimes we will restrict this and allow a detected mark's state to be chosen only from $\{0, 1\}$.

The last assumption is essentially the same as the Marking Assumption stated in section 2.4.5.

This means that a new object can be created, using more than one copy, by choosing in the detectable marks either of the two alternatives given, or an erasure. An erasure is any alternative different from the given two alternatives. In the rest of the object nothing is changed. If no erasures are made the result will be a copy that is close to the original. If there are erasures the result may or may not be close to the original. Note that if there is no collusion fingerprinting is trivial – the fingerprint assigned to a pirate will be readable in any illegal copy (s)he redistributes.

Embedding is the act of encoding '0's and '1's into the marks of an object. Note that any binary vector of length n can be used to choose between the alternatives in the marks in an object, and any object with n marks can define a binary vector of length n.

That the embedded code Γ is kept secret guarantees that the pirates do not know which mark corresponds to which public code coordinate, which public codeword corresponds to which user and how the bits in the code are represented in the object.

Using the terminology so far we can in words describe how the first part of the fingerprinting system (code choice and distribution) works:

1. The distributor has an original that he or she wants to distribute to M users.
2. The distributor chooses a public code using some performance measure, according to either the random code model or the known code model.
3. In the known code model the distributor, among the codes equivalent to the public code, chooses an embedded code, that is kept secret. In the random code model the embedded code is the same as the public code.
4. The distributor finds marks and embeds the codewords of the embedded code into the original, creating copies.
5. The distributor distributes the copies to the users.

2.4.5 Illegal Distribution

The illegal distribution is done by the pirates. The pirates will create a copy and distribute it illegally.

An *illegal copy* is a copy created by the pirates and illegally distributed. If the illegally created object is not close to the original the distributor does not care if it is distributed, since it is not a copy according to the definition.

The *illegal word*, z, is the binary vector, possibly with erasures, of length n that is embedded in the illegal copy.

The *strategy*, S, is the method that the pirates use to create the illegal copy. The pirates will in general include some random elements in S, either by choice or by necessity. The strategy will create the illegal copy in a way that the pirates think will best suite their purpose. A thorough description of pirate strategies can be found in chapter 6.

Definition 2.2: Let $\Gamma = \{\gamma^{(1)}, \gamma^{(2)}, \ldots, \gamma^{(M)}\}$ be an embedded fingerprinting code of length n and size M, and let $P = \{\gamma^{(i_1)}, \ldots, \gamma^{(i_c)}\}$ be the set of the pirates embedded codewords. The *feasible set* of P is defined as:

$$\mathrm{FS}(P, \Gamma) = \{y \in \mathrm{GF}(2)^n | y_j \in \{\gamma_j^{(i_1)}, \ldots, \gamma_j^{(i_c)}\}\}.$$

As we at any one time consider only one fingerprinting code, we will denote $\mathrm{FS}(P, \Gamma)$ by $\mathrm{FS}(P)$, and let the reference to the fingerprinting code be implicit. To save space we will sometimes use the notation $\tilde{P}$ for $\mathrm{FS}(P)$ in mathematical expressions.

The Marking Assumption: A collusion can only create objects containing fingerprints in the collusion's feasible set, possibly with some marks erased.

The feasible set is thus the set containing the words in the fingerprint space that is possible for the pirate collusion to construct – hence the name "feasible".

This means that what the strategy S does is to choose a word from the feasible set, according to some probability distribution. The choice of strategy is the choice among different probability distributions.

The Marking Assumption can be motivated by the fact that it is much easier for the pirates to make changes in places where their copies differ. Since they have no knowledge of the locations of the marks they will have to make many changes in order to be sure to actually change any undetectable marks. Doing so they risk making so much damage to the object that it is rendered useless. Thus the intuitive interpretation of the feasible set is as the set of words that a group of users can easily construct by "mixing" their copies.

2.4.6 Tracing and Testing

Extraction is the act of reading an embedded word from a copy. This word may contain erasures.

Known code model

The *tracing method*, *A*, is a method that takes the illegal word *z* as input and outputs a subset of the users. The performance of the tracing method is measured in the probability of not tracing the pirates correctly.

The distributor finds an illegal copy, extracts the illegal word and uses the tracing method to try to find one or more of the pirates. The users that the tracing method outputs are the ones that the distributor accuses of having taken part in the creation of the illegal word.

It is assumed that the pirates know the tracing method in relation to the public code, that is, how each possible vector is mapped to the set of user subsets. Using this the pirates will in general get partial information about the tracing method for the embedded code. The pirates' purpose is further assumed to be to prevent any of them from being accused, that is, they will use the strategy that minimizes the probability of this given all the information they have.

Below we will in words describe the second part of the fingerprinting system in the known code model, the part dealing with the illegal distribution and the tracing:

1. A set of pirates, *P*, using the strategy *S*, creates an illegal copy and distributes it illegally.
2. The distributor finds the illegal copy and extracts the illegal word, *z*.
3. The distributor uses the tracing method, *A*, and accuses the user subset that the tracing method outputs.

Random code model

The *testing method* is a method that takes a group of users and the illegal fingerprint as input and outputs either ‘yes’ or ‘no’ as to whether the group should be considered guilty.

The distributor finds an illegal copy, extracts the illegal word. If there is a group of users that is already suspected of being the pirate group, the testing method can be used to examine this.

Due to inherent problems with analysing the testing performance for groups that consist in part of pirates and in part of innocent users (see section 5.1), the performance of the testing method is measured in terms of the probability of accusing a group with only innocent users, and the probability of failing to accuse a group with only pirates.

It is assumed that the pirates know the testing method in relation to a random code in general, that is, how it is determined whether or not to accuse the group under consideration. The pirates’ purpose is further assumed to be to prevent any of them from being accused, that is, they will use the strategy that maximizes the method’s probability of failing to accuse a guilty group.

Below we will in words describe the second part of the fingerprinting system in the random code model, the part dealing with the illegal distribution and the testing:

1. A set of pirates, P, using the strategy S, creates an illegal copy and distributes it illegally.
2. The distributor finds the illegal copy and extracts the illegal word, z.
3. The distributor uses the testing method and tests whether a proposed group should be considered guilty or not.

2.4.7 Secrets in the Known Code Model

As we have seen, there are two secrets in the fingerprinting system: the choice of Γ among the codes equivalent to the public code Ψ and the positions of the marks in the object. These secrets are necessary for the fingerprinting system to work. If the positions of the marks are publicly known, pirates, or even a single pirate, can launch a very serious attack on the system by making changes in the object only in the marks. If Γ is known to the pirates they might be able to use that information to launch a stronger attack on the system than what is possible if Γ is secret. An example of this is the following.

Example 2.2: Suppose we have a fingerprinting code Γ with three codewords, $\gamma^{(1)} = (0\ldots00\ldots0)$, $\gamma^{(1)} = (0\ldots01\ldots1)$ and $\gamma^{(3)} = (1\ldots11\ldots1)$, where each block of bits $(0\ldots0$ or $1\ldots1)$ consists of δ bits. Thus we have that $n = 2\delta$. We assume that the holders of $\gamma^{(1)}$ and $\gamma^{(3)}$ are pirates. If Γ is known, the pirates can easily use $\gamma^{(1)}$ and $\gamma^{(3)}$ to create $\gamma^{(2)}$ by choosing from $\gamma^{(1)}$ in the first δ positions and from $\gamma^{(3)}$ in the rest. If Γ is not known there is a secret permutation of the coordinates that prevents the pirates from so easily creating $\gamma^{(2)}$. Instead, if they want to create this codeword, they have to guess which δ coordinates to choose from which codeword. If the coordinate permutation is chosen with uniform probability, the probability of creating $\gamma^{(2)}$ correctly, using $\gamma^{(1)}$ and $\gamma^{(3)}$, is then $1/\binom{2\delta}{\delta}$.

❑

2.4.8 Secrets in the Random Code Model

Just as in the known code model the positions of the marks have to be kept secret, and we also require that the users cannot in a detectable position tell which mark corresponds to ‘0’ and which corresponds to ‘1’. In the random code model there is no code structure so there is no need to use the construction with a public code that is different from the embedded code. However, the length, size and statistical properties are assumed to be known.

2.4.9 Model Discussion

Interesting problems are found where either the distributor or the pirates have a choice as to their actions. There are three such choices: the public code Ψ (which in the random code model is the same as the embedded code, and is not really *chosen*), the strategy S and the tracing/testing method. The code and the tracing/testing method are choices of the distributor, and the strategy is the pirates' choice. These three choices determine the probability of pirate success.

Most of the chapters in this thesis corresponds very well to these three choices. Chapter 6 is a discussion about how pirates can choose their strategy. Chapters 4 and 5 relate to both testing methods and pirate strategies. Chapter 8 is dedicated to coding problems. Chapter 7, about game theory, can be seen as linking everything as it explains how all the different choices together decide the final outcome.

From now on we will disregard the objects containing the embedded fingerprints and the users holding the objects. Instead we will talk only about the fingerprints (or codewords) without discriminating between them, the fingerprinted objects or the users. In the same way we are no longer interested in the marks and the mark positions, but in the bits and their positions (coordinates). The only thing that has to be kept in mind is that the pirates cannot tell the difference between ones and zeros, since the fingerprints are embedded in objects. The colluding pirates will only see that there are positions in which two or more objects differ.

We will also no longer discuss embedding techniques and the embedding problem, but simply assume that each fingerprinted object has embedded a binary vector γ of length n, the fingerprint, where each component γ_i corresponds to a mark in the object.

2.5 Performance of Fingerprinting Systems

Ideally we would like a fingerprinting system where the distributor never made any incorrect decisions when tracing or testing. Unfortunately this is not always possible to achieve in practice. If incorrect decisions are possible we must in some way be able to compare different systems and tell which is better and which is worse. To do this we need a performance measure.

It is natural to choose a probabilistic performance measure, for example measuring some kind of error probability for the testing or tracing. The "correct" choice of error probability would be the result of a full game theoretical analysis (since fingerprinting is fundamentally a game theoretical problem, as explained in chapter 7). Unfortunately such analyses are often, as in this case, not feasible to do due to complexity issues. What can be done then is to either find bounds on the error probability, or to simplify the problem in some way, for example by fixing, or disregarding, some otherwise variable property. An example is that by disregarding both the pirates' and the distributor's possibility to choose strategies, we can lose the dependence of everything but the properties of the code and get a strictly combinatorial performance measure. This kind of measure might not be as relevant as what comes out of a game theoretical analysis, but on the other hand it is simple enough to handle, and it is much more suitable for finding code constructions.

A special example of simplification is the way that Boneh and Shaw handle the problem in [BS98]. They construct a code with properties such that the effects of the pirates' strategies are so limited that it is very simple to analyse the resulting system.

2.6 Connection to Error Control Coding

The fingerprinting model we use is basically a coding theoretical one. We consider the set of fingerprints to be a binary code of length n and size M. We can also choose to view the fingerprinting problem as a communication problem that uses channel coding or error control coding. One or several codewords are chosen and transmitted. These transmitted codewords correspond

to the fingerprints belonging to the colluding pirates. The transmission is not perfect, so only one word is received, the illegal fingerprint, and this is the result of some kind of unknown "mix" of the transmitted codewords. The received word is then used to estimate which codeword or codewords were transmitted. In the communication situation, this scenario is called "multiple access" and the act of determining which codewords were sent, using the received word, is called decoding.

2.6.1 Distance Properties

The distance between two vectors can be defined as the number of coordinates that differ between them, called the Hamming distance. As it is, however, the Hamming distance does not say very much in the case of binary fingerprinting. Intuitively, and in analogy to error control coding, one might think that it would be better if the different fingerprints were well separated, that is, had a large Hamming distance. That this is not always the case can be seen in the following examples.

Example 2.3: There exist linear codes with very good distance properties, but the linearity itself is a problem for fingerprinting codes, as will be shown. Assume that we are using a linear fingerprinting code, Γ, and that three users with fingerprints $\gamma^{(1)}, \gamma^{(2)}, \gamma^{(3)} \in \Gamma$ are colluding. The pirates create a new object containing the fingerprint z by making a minority decision in all detectable positions. Since there are exactly three pirates, and since the undetectable positions will be output unaltered to z, this is the same as creating z as the modulo-2 sum of $\gamma^{(1)}$, $\gamma^{(2)}$ and $\gamma^{(3)}$. Because of the linearity this new fingerprint z is another codeword in the code and may already be assigned to some other, innocent user. The three pirates have thus easily created a new fingerprint that will frame an innocent user, something that, of course, is not acceptable. This property of linear codes for fingerprinting has been reported in [TL97].

❑

Example 2.4: Consider a fingerprinting code in which both the all-zero codeword $\bar{0}$ and the all-one codeword $\bar{1}$ are members. Suppose that two pirates have received these fingerprints in their fingerprinted objects, and that they want to create an illegal copy together. To them all positions are detectable (which is observable to them in both code models), and they can thus create any fingerprint, without restriction. A possible strategy for them would be to flip a coin for each position and, depending on whether the coin shows heads or tails, they choose to output the contents of either the first or the second pirate's fingerprint. This way the illegal copy's fingerprint will be completely random, and thus untraceable. Of course this strategy works equally well for all cases where the pirates' fingerprints are the inverses of each other.

❑

Example 2.4 shows that the extreme case when two pirates' codewords are as far as possible from each other, in the Hamming distance sense, is very bad for the distributor and very good for the pirates. Obviously we do not want the pirates' codewords to be too far away from each other. On the other hand, if the codewords of the fingerprinting code are very close to each other, it suffices to make small alterations of the pirates' fingerprints for the pirates to be able to frame somebody outside the collusion, somebody innocent. This is also undesirable. Thus we do not want the codewords of the fingerprinting code to be too close to each other (in the Hamming sense) either.

What these examples, and the discussion above, show is that the Hamming distances within the fingerprinting code have no straight forward relationship with how good the code is for fingerprinting. Some other kind of measure is needed to be able to evaluate fingerprinting codes. One example of another measure is the one presented in chapter 8. We will however find that in the bounds we construct in chapters 3 and 8 there are some connections between fingerprinting performance and Hamming distances.

2.7 The Problem

2.7.1 Fingerprinting Goals

Both when trying to find the pirates and when testing whether a certain, proposed group of users is guilty, we are really making a hypothesis test, and it is thereby possible to make two kinds of error. The false alarm error corresponds to accusing a user that is innocent and the false dismissal error corresponds to failing to accuse a user that is guilty. It is desirable to keep the probability of both of these errors small, but unfortunately they are conflicting goals, so a trade-off has to be made. An interesting special case is when exactly one user is accused after every tracing. Then the two kinds of error coincide. A false alarm is always a false dismissal and vice versa. This way we get only one kind of error.

As can be expected, it is possible to get lower error probabilities if longer fingerprints are used. The problem is that for every object there is a practical limit on how large a fingerprint can be embedded. Therefore the smaller the fingerprints we can cope with for a given error probability, the better. This reasoning holds also when other performance measures than error probability are chosen.

Another thing that affects the error probability is the number of colluding pirates. A small limit on the number of colluding pirates that we can handle means that it is easier for the pirates to get such a number together. Thus we want to be able to handle as many colluding pirates as possible.

In summary, our goals are:

- to get as good performance (for example, low error probability) as possible when finding or testing pirates
- to use as small fingerprints as possible
- to handle as many colluding pirates as possible

These are, of course, conflicting goals, and what we will try to do in the thesis is to find numerical relations between them, both as theoretical limits, and in explicit systems.

2.7.2 General Problem and Restrictions

The goal of this thesis is to try to answer the following questions regarding fingerprinting of digital data:

- How can fingerprinting systems be analysed?
- What kind of performance can we get in fingerprinting systems?
- How can we construct a working fingerprinting system?

The first question is examined in chapters 6 and 7, where the possible strategies of the pirates are described, and a game theoretical description of the problem is made. This description shows that it is not feasible to find the general game theoretical solution to the problem, why chapter 8 introduces a simplified performance measure, dependent only on the code, to make it possible to attack the problem of analysing a fingerprinting system.

The second question is discussed in chapter 3, where two simple bounds on performance are introduced. Bounds and constructions are also described in chapter 8, for the performance measure introduced there. In chapter 5 a simple testing method is described, and its error probability is derived.

The third question is the least discussed one. Chapter 4 describes a testing method that does *not* work, unless the whole pirate group is tested. Chapter 5 uses this result and describes a simple way to test whether a proposed group should be considered guilty.

Chapter 3

Bounds on Fingerprint Length

This chapter gives two bounds on the performance of fingerprinting systems. The first bound is general and the second is dependent on what fingerprinting code is used. These bounds are in no way a complete set of all interesting bounds, but rather what we have been able to derive.

Optimally, we would like to have upper and lower bounds on the fingerprint length for both testing methods and methods for finding pirate collusions (tracing methods). What we have is two bounds. One is a lower bound on the fingerprint length needed for it to be possible to accuse every individual collusion of at most size c among M users. The bound does not take into account the degree to which such accusations are correct, only that they are possible.

The other bound is a lower bound on the probability of failing to find one of the pirates under an assumption on the tracing method. For this bound to be usable, the size of the feasible set has to be possible to calculate given the size of the collusion.

3.1 A Bound for Collusions

3.1.1 Introduction

When tracing the collusion of pirates given an illegal fingerprint z, we make a partition of the fingerprint space into disjunct subsets, where each subset corresponds to a certain set of users to accuse of being the colluding pirates. In principle, finding the collusion is then just a matter of checking which subset of the fingerprint space z belongs to. This is what makes up the tracing method. Of course, in practice this partitioning will probably not be explicit, but rather the implicit result of some tracing algorithm.

Assume that given an illegal fingerprint z we want to be able to find any collusion of at most c pirates among the total number of M users. For this to be possible there must be at least as many possible fingerprints as there are possible collusions of size at most c. The bound we are going to derive is based on the idea that different collusions must be able to generate different fingerprints in order to be distinguishable. This tells us nothing about the performance of the tracing method, but it does constitute a lower bound on the size of the fingerprint space necessary for it at all to be possible to have a deterministic tracing method able to accuse any collusion of at most size c.

The bound requires exactly one unique fingerprint for each collusion, while in fact most collusions have a considerable freedom in their choice of fingerprints (that is, their feasible sets are large). Consequently, the actual fingerprint length required is expected to be much greater than this lower bound.

3.1.2 The Bound

If the total number of users is M, the number of possible user sets of size at most c is

$$\sum_{i=1}^{c} \binom{M}{i} = \sum_{i=0}^{c} \binom{M}{i} - 1 = V_M(c) - 1\,, \tag{3:1}$$

where $V_M(c)$ is the volume of a sphere of radius c in an M-dimensional Hamming space. This means that there have to be $\lceil \log_2(V_M(c)-1) \rceil$ bits in the fingerprint for it not to be impossible to uniquely accuse each of the possible user sets.

In the rest of this chapter we will use the following definition, which is the binary entropy function.

Definition 3.1: $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$

From theorem A.3.12 in [SR92] we get the following inequalities regarding the volume of a binary M-dimensional sphere or radius c:

$$\frac{1}{\sqrt{8c(1-(c/M))}} \times 2^{MH(c/M)} < V_M(c) \le 2^{MH(c/M)} \text{ if } \frac{c}{M} \le \frac{1}{2}. \qquad (3:2)$$

3.1.3 The Length of Fingerprints

The number of different user sets that we have to be able to point out is one less than the volume of an M-dimensional sphere with radius c. Thus we can say that to be able to identify any user set of at most size c, the length n of the fingerprints has to fulfil the following inequality:

$$n \ge \log(V_M(c)-1) > \log\left(\frac{V_M(c)}{2}\right) > \log\left(\frac{1}{2\sqrt{8c(1-(c/M))}} 2^{MH(c/M)}\right)$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) + MH\left(\frac{c}{M}\right)$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) - c\log\frac{c}{M} - M\left(1-\frac{c}{M}\right)\log\left(1-\frac{c}{M}\right) =$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) - c\log c + c\log M - M\log\left(1-\frac{c}{M}\right) + c\log\left(\frac{M-c}{M}\right)$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) - c\log c - M\log\left(1-\frac{c}{M}\right) + c\log(M-c)$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) - c\log c - \frac{M\ln\left(1-\frac{c}{M}\right)}{\ln 2} + c\log(M-c) \quad (3:3)$$

In expression (3:3) above, $\ln(x)$ means the natural logarithm, and we can thus bound the expression using the well-known inequality $\ln(1-x) \le -x$. This means that:

$$n > -\frac{1}{2}\log(32c(1-(c/M))) - c\log c - \frac{M\ln\left(1-\frac{c}{M}\right)}{\ln 2} + c\log(M-c)$$

$$\ge -\frac{1}{2}\log(32c(1-(c/M))) - c\log c + \frac{c}{\ln 2} + c\log(M-c)$$

$$= -\frac{1}{2}\log(32c(1-(c/M))) - c\log c + c\log e + c\log(M-c)$$

$$= -\frac{1}{2}\log 32c - \frac{1}{2}\log(1-(c/M)) + c\log(M-c) - c\log c + c\log e$$

$$= -\frac{1}{2}\log 32c - \frac{1}{2}\log(1-(c/M)) + c\log\left(\frac{e(M-c)}{c}\right). \quad (3:4)$$

We can see that (3:4) grows approximately logarithmically in M. Studying it to see how the expression is affected by c, we see that the last term is the dominating one. c is upper bounded by M, and will in most cases be much smaller, so for small values of c, (3:4) grows approximately linearly in c. The bound is plotted below for some different values of c (number of pirates in the collusion) and some different values of M (total number of users).

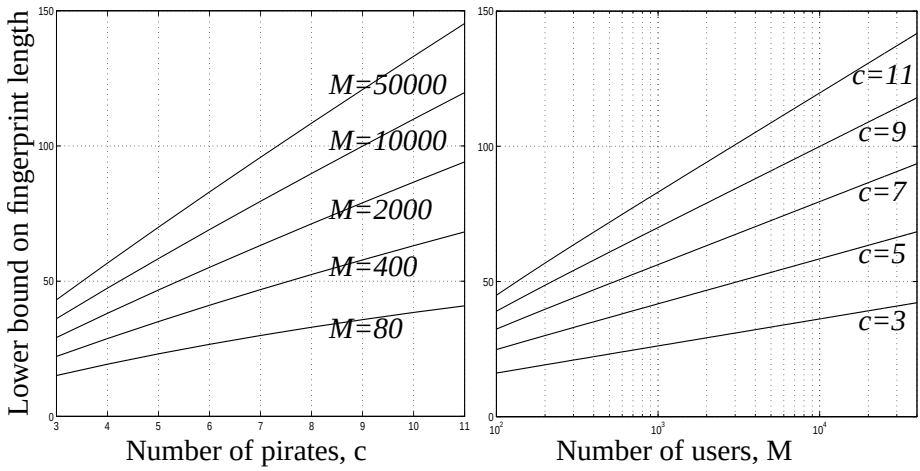

Figure 3.1: *The lower bound on the fingerprint length needed as a function of c (left) and M (right). The different curves are for different values of the other, fixed, variable.*

In figure 3.1 we see that, at least in the plotted range, the approximation that the lower bound on the fingerprint length needed grows logarithmically in M and linearly in c, is quite good.

The bound in this section is not very strongly connected to what is done in other chapters in this thesis. It is a simple construction using the assumptions that the tracing method accuses an entire group of users, and that no group of users will enjoy impunity due to the fact that the tracing method never accuses the group, regardless of what illegal fingerprint is found.

Under these assumptions we have found a lower bound on the needed fingerprint length as a function of the collusion size and the number of users in the

system. The needed fingerprint length is more dependent of the collusion size than the number of users in the system, so it seems that it is the collusion size that is the limiting factor in a fingerprinting system like this.

3.2 A Bound for a Single Pirate in a Collusion

3.2.1 Derivation of the Bound

For the next bound we will change our goal a little. Instead of requiring the ability to accuse any collusion, we will instead assume that the tracing algorithm outputs exactly *one* user. It should be noted that the construction of this bound is similar to the construction of the upper bound in section 8.6.2.

Much as in section 3.1, the idea is that the tracing method implies a partitioning of the entire fingerprint space, which we denote U, into M disjoint subsets, each corresponding to a certain user to be accused (M is the total number of users). We call these sets *accusation sets* and denote the accusation set of user i, A_i. When we find an illegal fingerprint, we check which accusation set A_i the fingerprint belongs to and accuse the corresponding user i of having taken part in the creation of the illegal fingerprint.

When a group of pirates, P, collude to make an illegal copy, they can choose the illegal fingerprint z from their feasible set $\tilde{P} = \mathrm{FS}(P)$ (the feasible set is the set of fingerprints that is possible for the pirates to create under the Marking Assumption; see chapter 2). What we would like is the feasible set of the collusion and the union of the pirates' accusation sets to overlap as much as possible. Preferably the pirates' feasible set would be a subset of the union of the pirates' accusation sets. This may not be possible in general so we will accept that a fraction of the words in the pirates' feasible set will not be part of the union of the accusation sets of the pirates; but the smaller this fraction is, the better.

We assume that the tracing method gives the partitioning of the fingerprint space U the property that all accusation sets A_i are upper bounded in size, so that $|A_i| \leq (k|U|)/M$, for some $k \geq 1$ (with $|A_i|$ denoting the cardinality of

the set A_i). If all accusation sets are the same size, then we can choose $k = 1$. We think it is reasonable to assume that for most tracing methods k is not very large.

Let A be the union of the colluding pirates' accusation sets and let $\varepsilon_P = \left|\tilde{P} \setminus A\right| / \left|\tilde{P}\right|$ be the fraction of $\tilde{P}$ not in the union of the colluding pirates' accusation sets A. (With $\tilde{P} \setminus A$ we mean $\tilde{P} \cap A^*$, where A^* denotes the complement of the set A). c is the number of pirates in the collusion. Under these conditions $|A| = \sum_i |A_i| \leq (ck|U|)/M$, and we get the bound

$$\varepsilon_P = \frac{\left|\tilde{P} \setminus A\right|}{\left|\tilde{P}\right|} \geq \frac{\left|\tilde{P}\right| - |A|}{\left|\tilde{P}\right|} = 1 - \frac{|A|}{\left|\tilde{P}\right|} \geq 1 - \frac{ck|U|}{\left|\tilde{P}\right|M}. \tag{3:5}$$

ε_P is the fraction of the feasible set $\tilde{P}$ that does not accuse any pirate in the collusion, and because of this we would like to keep it small. If we do not, the pirates can use the simple attack on the fingerprinting scheme consisting of choosing randomly among the fingerprints in their feasible set. Doing so, they will have a probability ε_P of succeeding in creating an illegal fingerprint that will cause somebody outside the collusion to be accused. This means that when the pirates create the illegal fingerprint in this way, the fraction ε_P is actually the error probability. This way of creating the illegal fingerprint is not necessarily a very good one in general for the pirates, but the point is that the fingerprint length needed to handle this attack will be a lower bound on the fingerprint length needed to handle the pirates' most efficient attack.

To be able to analyse expression (3:5) we must know the size of the pirates' feasible set, $\left|\tilde{P}\right|$. This is only possible if we know the code being used. How this can be used for the random code in the random code model will be explored next.

3.2.2 Independent, Random Code

Consider the fingerprinting code in the random code model, where each fingerprint is made up of random bits, each drawn independently and with the probability $p = 1/2$ of drawing a '1'.

We will now mimic expression (3:5), with the exception that we will use expected values instead, since we are dealing with a random code.

$$\mathrm{E}[\varepsilon_P] = \mathrm{E}\left[\frac{\left|\tilde{P} \setminus A\right|}{\left|\tilde{P}\right|}\right] \geq \mathrm{E}\left[\frac{\left|\tilde{P}\right| - |A|}{\left|\tilde{P}\right|}\right]$$

$$= \mathrm{E}\left[1 - \frac{|A|}{\left|\tilde{P}\right|}\right] = \mathrm{E}\left[1 - \frac{ck|U|}{\left|\tilde{P}\right|M}\right] = 1 - \frac{2^n ck}{M}\mathrm{E}\left[\frac{1}{\left|\tilde{P}\right|}\right] \qquad (3:6)$$

To see how the lower bound (3:6) behaves we have to find an expression for $\mathrm{E}[1/\left|\tilde{P}\right|]$.

Given the code under consideration, the probability of a certain position being detectable is $1 - 2^{1-c}$, which is the probability that not all the c bits in a certain coordinate are the same. Let X be the random variable denoting the number of positions that will be detectable. X is then a binomially distributed random variable with parameters $\alpha = 1 - 2^{1-c}$ and n.

With this new notation we are ready to return to the problem of finding an expression for $\mathrm{E}[1/\left|\tilde{P}\right|]$.

$$\mathrm{E}\left[\frac{1}{\left|\tilde{P}\right|}\right] = \mathrm{E}[2^{-X}] = \sum_{j=0}^{n} \binom{n}{j} \alpha^j (1-\alpha)^{n-j} 2^{-j}$$

$$= \sum_{j=0}^{n} \binom{n}{j} \left(\frac{\alpha}{2}\right)^j (1-\alpha)^{n-j} = \left(\frac{\alpha}{2} + (1-\alpha)\right)^n$$

$$= \left(1 - \frac{\alpha}{2}\right)^n \qquad (3:7)$$

Inserting (3:7) in (3:6), using that $\alpha = 1 - 2^{1-c}$, we get:

$$E[\varepsilon_P] \geq 1 - \frac{2^n ck}{M}\left(1 - \frac{\alpha}{2}\right)^n = 1 - \frac{2^n ck}{M}\left(1 - \frac{1 - 2^{1-c}}{2}\right)^n$$

$$= 1 - \frac{2^n ck}{M}\left(\frac{1 + 2^{1-c}}{2}\right)^n = 1 - \frac{ck}{M}(1 + 2^{1-c})^n . \qquad (3:8)$$

To get an idea of how expression (3:8) behaves, we plot it as a function of *n*, and for different values of *c* and *M*. We have chosen $k = 1$ in the figures below. The effect of *k* is exactly the inverse of the effect of *M*, so it is relatively easy to estimate the effects of using another *k*.

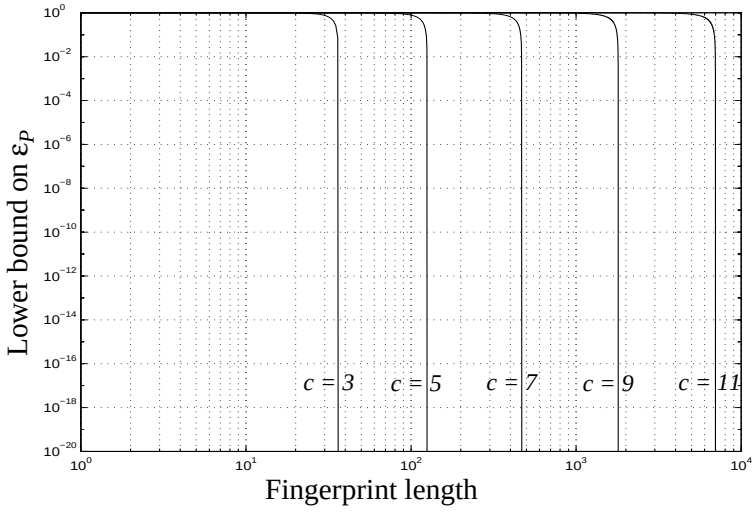

Figure 3.2: *The lower bound on ε_P for different number of pirates, c. The number of users in the system is M=10000, and the constant k=1.*

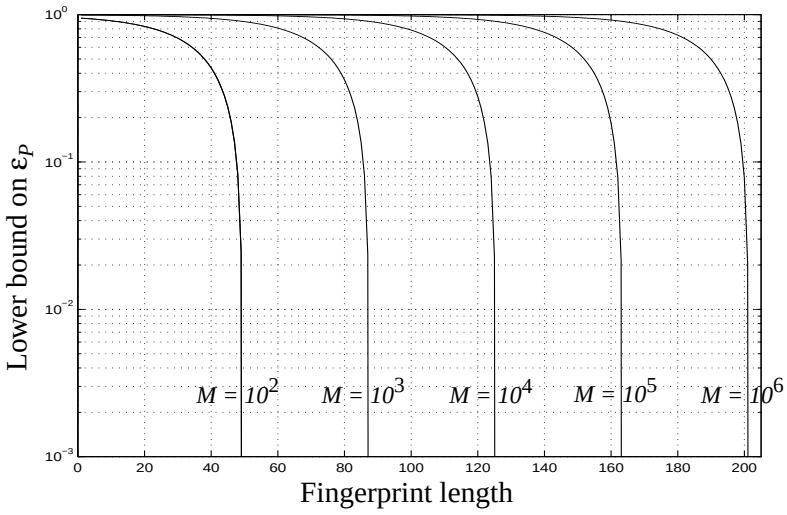

Figure 3.3: *The lower bound on ε_P for different total number of users, M. The number of pirates is c=5 and the constant k=1.*

Examining expression (3:8) we can see that the curves in figures 3.2 and 3.3 tend not towards zero with growing n but towards $-\infty$.

From the figures we can see that the fingerprint length needed to keep the lower bound fixed seems to grow exponentially in c and logarithmically in M. To make a more thorough analysis of this we require the lower bound to be less than zero and see how n depends on c and M when $k = 1$. This way we know that we do not force the expected value of ε_P to any value greater than zero. Also, since the lower bound in (3:8) drops so steeply towards, and past zero, the value we require the bound to be lower than, will affect the required fingerprint length very little.

We substitute $\mathrm{E}[\varepsilon_P]$ with 0 in (3:8) and get $0 \geq 1 - (ck/M)(1 + 2^{1-c})^n$. This implies that:

$$n \geq \frac{\log M - \log ck}{\log(1 + 2^{1-c})} \geq \frac{\log M - \log ck}{2^{1-c}\log(e)} = 2^c\left(\frac{\log M - \log ck}{2\log(e)}\right). \qquad (3:9)$$

In the last step we have used the inequality $x \geq \ln(1 + x)$. We see that the lower bound in (3:9) grows logarithmically in M and exponentially in c.

3.3 Discussion

Even though the bounds in sections 3.1 and 3.2 are bounds for different things, and hence not really comparable, it is interesting to see that they have some properties in common.

Let us first summarize our results from section 3.1 and 3.2. The bound in section 3.1 grows approximately logarithmically in M and linearly in c, and the bound in section 3.2 grows approximately logarithmically in M and exponentially in c. What is similar is that both bounds grow logarithmically in M and that they both grow faster in c than in M.

Chapter 4

Correlation-Based Testing

This chapter is devoted to questions about correlation-based testing methods in the random code model. With testing we mean the act of using the fingerprint of an illegal copy to test whether a proposed group of users should be considered guilty or not.

The contents of this chapter are inspired by the work by Siegenthaler in [TS84], where the concept of correlation-immunity is introduced. There it is used for analysing combining functions for stream ciphers. We will use it in much the same way, but in our case the input will be fingerprints of pirates in a collusion and the output will be the illegal fingerprint they create, that is, the combining function corresponds to the pirate strategy.

4.1 Introduction

It would be of great help to the distributor if it were possible to test whether a certain group of users is a subset of the whole guilty group. In that case, if the distributor suspects some of the users (s)he could test to see if they are guilty, without having to care about whether they constitute the whole collusion or not.

Example 4.1: Suppose that $\gamma^{(1)}, \ldots, \gamma^{(5)}$ are the fingerprints of the five users in a fingerprinting scheme, and that users number one, two and three are pirates. The pirates collude and make a new, illegal copy z by choosing for all detectable positions i the bits from $\gamma^{(1)}$ with probability 1/3, the bits from $\gamma^{(2)}$ with probability 1/3 and the bits from $\gamma^{(3)}$ with probability 1/3. This means that approximately one third of the bits in the detectable positions will be chosen from each of the pirates' fingerprints.

Let us examine the probability of ones in z among the positions in which $\gamma^{(1)}$ contains a '1', that is, $\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 1)$. Given that $\gamma_i^{(1)} = 1$ z will contain a '1' in position i if any of the following is true:

1. the output is chosen from $\gamma^{(1)}$
2. the output is chosen from $\gamma^{(2)}$ and $\gamma_i^{(2)} = 1$
3. the output is chosen from $\gamma^{(3)}$ and $\gamma_i^{(3)} = 1$

This probability can be calculated as:

$$\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 1)$$

$$= \Pr(\text{choose from } \gamma^{(1)}) + \Pr(\text{choose from } \gamma^{(2)}) \times \Pr(\gamma_i^{(2)} = 1) +$$

$$+ \Pr(\text{choose from } \gamma^{(3)}) \times \Pr(\gamma_i^{(3)} = 1)$$

$$= \frac{1}{3} + \frac{1}{3} \times \frac{1}{2} + \frac{1}{3} \times \frac{1}{2} = \frac{2}{3}.$$

In much the same way we can get:

$$\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 0) = 0 + \frac{1}{3} \times \frac{1}{2} + \frac{1}{3} \times \frac{1}{2} = \frac{1}{3}.$$

In this case the result would have been the same regardless of which pirate's fingerprint we used to point out the positions for which we calculated the probability of ones. For the users not in the collusion that created the illegal fingerprint z, both of these probabilities would have been 1/2. This means that given a user's fingerprint and the illegal fingerprint, it is possible to make a hypothesis test as to whether these probabilities can be considered to be different from 1/2. As the probabilities are 1/2 if the user is innocent, we will suspect the user of having been a part of the collusion if we can say with a high reliability that the probabilities are not 1/2.

❑

This example shows that it is sometimes possible to test whether a specific, single pirate has been part of the collusion that has created an illegal fingerprint z. In these cases it is easy to find the whole collusion by just doing this test for every user in the system. The test can be done so that the probability of accusing an innocent user is arbitrarily small, although the smaller the probability of false accusation, the greater the probability of false dismissal. That it is not always possible to find pirates in this way is shown in the next example.

Example 4.2: Suppose that $\gamma^{(1)}, \ldots, \gamma^{(5)}$ are the fingerprints of the five users in a fingerprinting scheme, and that users number one, two and three are pirates. The pirates collude and make a new, illegal copy z by making a minority decision for all detectable positions, that is, they output to z that pirate's bit value that is different from the two others.

Let us examine the probability of ones in z among the positions i in which $\gamma^{(1)} = 1$, that is, $\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 1)$. In such positions $z_i = 1$ if either of the following is true:

1. position i is undetectable, that is, $\gamma_i^{(2)} = \gamma_i^{(3)} = 1$.
2. position i is detectable and $\gamma_i^{(2)} = \gamma_i^{(3)} = 0$.

This probability can be calculated as:

$$\begin{aligned}
&\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 1) \\
&= \Pr(\gamma_i^{(2)} = \gamma_i^{(3)} = 1) + \\
&+ \Pr(\gamma_i^{(2)} = \gamma_i^{(3)} = 0) \\
&= \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.
\end{aligned}$$

In much the same way we can get:

$$\Pr(z_i = 1 \,|\, \gamma_i^{(1)} = 0) = 1 - \left(\frac{1}{4} + \frac{1}{4}\right) = \frac{1}{2}.$$

Here the result is that the individual pirates as well as the users get the probabilities 1/2. This means that the kind of hypothesis test that worked so well in example 4.1 is useless here. The pirates and the innocent users look the same in this sense, so we cannot find the collusion this way.

❑

The kind of hypothesis test mentioned will, in practice, consist of computing the ratio of ones in the positions of the illegal fingerprint, for which a certain user's fingerprint is '1', and the ratio of ones in the positions of the illegal fingerprint, for which the same user's fingerprint is '0'. If these ratios differ from the expected 1/2 we say that there is a correlation between the two binary vectors (in this case, the user's fingerprint and the illegal fingerprint). This can be generalized to more than one user. We can just as well compute

the ratios of ones in the positions of the illegal fingerprint, for which two users' marks are '00', '01', '10' and '11'. For larger groups of users the computations are done analogously.

It is always possible to try this kind of hypothesis testing, because if the user, or users, under consideration are not part of the collusion, the probabilities of ones are always equal to 1/2. The problem is, as we saw in example 4.2, that it will not always work. Whether it works or not depends on the strategy the pirates in the collusion used when they created the illegal fingerprint. In the following sections we will make a systematic study of what strategies the pirates can use if they want it to be impossible to find them in this way.

As will be seen later (lemma 4.1), it is always possible to use correlation-based methods on the whole group of pirates. This is so because of the pirates' lack of choice of what to output in the undetectable positions.

4.1.1 A Model for Pirate Strategies

The fingerprints used consist, according to the random code model, of bits chosen randomly and independently, with an equal probability of '1' and '0'. We will model the pirates' strategy when creating an illegal fingerprint as that they combine their individual fingerprints according to some function, and that the result of this function is the new, illegal fingerprint z. The combining function will be assumed to be memoryless, that is, it works independently in each coordinate. To make it easier to talk about the contents of a set of fingerprints in a certain position we will make a definition.

Definition 4.1: If $\langle \gamma^{(i_1)}, \ldots, \gamma^{(i_c)} \rangle$ is a tuple of fingerprints, all different, then $x^{(j)} = (\gamma_j^{(i_1)}, \ldots, \gamma_j^{(i_c)})$ is the jth *input vector* of this tuple.

The jth input vector of a tuple of fingerprints is thus the vector consisting of the bits in the jth coordinates of the fingerprints in this tuple.

So far this is very similar to [TS84], but we will allow more general combining functions than those used there. We will assume that the pirates, for each different value on the input vector, choose a probability that the output will be a '1'. In mathematical notation they will choose a function

$f(\{0,1\}^c) \rightarrow \Re[0,1]$ that operates on an input vector, and yields as output the probability that the corresponding bit in the illegal fingerprint will be a '1'. The pirates cannot choose this function, *f*, completely at will. For example, according to section 2.4.8, the pirates cannot tell the difference between ones and zeros when they are embedded, a fact that imposes certain restrictions on *f*. In this model the deterministic functions used in [TS84] constitute the special case when only the probabilities 0 and 1 are allowed as results of the function *f*.

Example 4.3: Consider three pirates with fingerprints $\gamma^{(1)}$, $\gamma^{(2)}$ and $\gamma^{(3)}$. The pirates want to create an illegal fingerprint using the model above and choose *f* in the following way:

$x^{(i)}$	$f(x)$
(000)	0
(001)	0.4
(010)	0.7
(011)	0.5
(100)	0.5
(101)	0.3
(110)	0.6
(111)	1

Table 4.1: *Function table for the pirates' combining function f.*

Two things are important to notice here:

$f(000) = 0$ as the pirates have no choice in this case. The probability of one in the output has to be zero, since such positions are undetectable and therefore unchangeable for the pirates. $f(111) = 1$ for the same reason.

$f(001) = 1 - f(110)$. The reason for this is that the pirates cannot tell the difference between (001) and (110) when the input vectors are embedded in the data, so they will act in the same way in both of these cases. But as they do not know which embedded value means '0' and which means '1' they can only choose whether or not to output the embedded value found in one, specific fingerprint of theirs. (What the pirates see is that fingerprint number one and two contain the same data, and that fingerprint number three differs.) The input vectors in the two cases are the inverses of each other, which means the probability of outputting a '1' in one case is the same as the probability of outputting a '0' in the other case. This yields $f(001) = 1 - f(110)$. The same argument shows that the pirates must choose $f(010) = 1 - f(101)$ and $f(011) = 1 - f(100)$.

This means that the pirates cannot really make a combining function like that in table 4.1 with the probabilities chosen freely, but instead their function table will look something like table 4.2 below.

In table 4.2, X and Y mean different embedded values which the pirates cannot tell whether they mean '0' or '1'.

$x^{(i)}$	$\Pr(z_i = Y)$
(XXY)	0.4
(XYX)	0.7
(XYY)	0.5

Table 4.2: *The pirates' own combining function table, corresponding to f.*

❑

4.2 Correlation-Immune Combining Functions

As discussed previously, it would be of great help to the distributor if it was possible to test whether a certain group of users is a subset of the whole guilty group. As it would be of help to the distributor to be able to do this, it would benefit the pirates if they could prevent it. Thus the pirates would like the combining function to be such that the distributor is not able to learn from the illegal fingerprint z any information about the fingerprints of any true subset of the pirates. In information theoretical terms, the property that the distributor should not be able to learn from the illegal fingerprint z any information about the fingerprints of any true subset of the pirates is then equivalent to:

$$I(x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_m}^{(i)}; z_i) = 0 . \tag{4:1}$$

where $m < c$, $I(.,.)$ means the mutual information, $x_{j_1}^{(i)}, \ldots, x_{j_m}^{(i)}$ is the input vector of a true subset of the pirates, and z_i is the ith bit in the illegal fingerprint z. (4:1) is equivalent to z_i being statistically independent of any m-tuple obtained by choosing m components $x_{j_1}^{(i)}, \ldots, x_{j_m}^{(i)}$ from the input vector $(x_1^{(i)}, \ldots, x_c^{(i)})$, $m < c$. This property is what is called correlation-immunity in [TS84].

Note that z is a function of random variables and thereby also random, even in the case when the combining function is deterministic. This means that expression (4:1) is meaningful even when the combining function f only takes values 0 or 1, that is, when it is deterministic.

Definition 4.2: $f(x^{(i)})$ is mth-order *correlation-immune* if every t-tuple obtained by choosing t components from the input vector $x^{(i)}$ is statistically independent of z_i for all $t \leq m$.

This is equivalent to:

$$\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y) = \Pr(z_i = 1) \text{ for every choice of } j_1, \ldots, j_t \in [1, c], \text{ all different, and any } y \in \{0, 1\}^t, \text{ for all } t \leq m. \quad (4:2)$$

Lemma 4.1: Under the Marking Assumption, there is no combining function $f(\{0, 1\}^c) \rightarrow \Re[0, 1]$ that is *c*th order correlation-immune.

Proof: Due to the Marking Assumption, the function *f* must for all undetectable positions containing only ones, yield '1' as output, that is, the probability $\Pr(z_i = 1 | x^{(i)} = (1 \ldots 1)) = f(1 \ldots 1) = 1$. Since the probability of input vectors $x^{(i)} = (0 \ldots 0)$ is not zero, and since $f(0 \ldots 0) = 0$, we get that the unconditional probability of ones in *z*, $\Pr(z_i = 1) \neq 1$, that is, $1 = \Pr(z_i = 1 | x^{(i)} = (1 \ldots 1)) \neq \Pr(z_i = 1)$, and f is not *c*th order correlation-immune.

❑

Because of lemma 4.1 we make the following definition:

Definition 4.3: A combining function $f(\{0, 1\}^c) \rightarrow \Re[0, 1]$ is called *maximally correlation-immune* if it is (*c*-1)th order correlation-immune.

This means that the way for the pirates to create a fingerprint that does not give any information about any true subset of the collusion is to create the illegal fingerprint using a combining function that is maximally correlation-immune. That way the probability of ones in the positions specified by the input vectors of true subsets of the collusion will be equal to the probability of ones in the entire illegal fingerprint *z*. In other words, it will be impossible to find a true subset of the whole guilty group with correlation based methods.

As stated earlier, the pirates cannot choose this function, *f*, completely at will. Restrictions in their freedom to choose their combining function will be modelled as restrictions on the function *f* when the analysis is made of how the pirates can achieve an illegal copy in a way that is maximally correlation-immune.

4.2.1 Maximally Correlation-Immune Combining Functions

What we want to do now is to examine what the function *f* must look like if we are to get maximum correlation-immunity. This means that in the following we will assume that $m = c - 1$, where c is the number of inputs (pirates in our case).

Lemma 4.2: If $f(x)$ is a maximally correlation-immune combining function and $w_H(x') = w_H(x'')$, where $w_H(.)$ is the Hamming weight, then $f(x') = f(x'')$.

Proof: We examine expression (4:2) under the assumption that $t = m = c - 1$ and with $x_{j_q}^{(i)} = 0$ for all j_q except one, the sth bit, which is free.

$$\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})$$

$$= \frac{\Pr((z_i = 1) \cap ((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0}))}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})}$$

$$= \frac{\Pr((z_i = 1) \cap ((x^{(i)} = \bar{0}) \cup (x^{(i)} = 0 \ldots 010 \ldots 0)))}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})},$$

where $\bar{0}$ is the vector consisting of only zeros.

As $x^{(i)} = \bar{0}$ and $x^{(i)} = (0\ldots010\ldots0)$ are disjoint events, this is equal to:

$$\frac{\Pr((z_i = 1) \cap (x^{(i)} = \bar{0})) + \Pr((z_i = 1) \cap (x^{(i)} = 0\ldots010\ldots0))}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})}$$

$$= \frac{\Pr(x^{(i)} = \bar{0})\Pr(z_i = 1 | x^{(i)} = \bar{0})}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})} +$$

$$+ \frac{\Pr((x^{(i)} = 0\ldots010\ldots0))\Pr(z_i = 1 | (x^{(i)} = 0\ldots010\ldots0))}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})}$$

$$= \frac{\Pr(x^{(i)} = \bar{0})}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})} f(\bar{0}) +$$

$$+ \frac{\Pr(x^{(i)} = 0\ldots010\ldots0)}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_{c-1}}^{(i)}) = \bar{0})} f(0\ldots010\ldots0)$$

$$= \frac{2^{-c}}{2^{-(c-1)}} f(\bar{0}) + \frac{2^{-c}}{2^{-(c-1)}} f(0\ldots010\ldots0)$$

$$= \frac{1}{2} f(0\ldots0) + \frac{1}{2} f(0\ldots010\ldots0). \qquad (4:3)$$

For it to be possible for f to be maximally correlation-immune, (4:3) must be equal to $\Pr(z_i = 1)$:

$$\frac{1}{2} f(0\ldots0) + \frac{1}{2} f(0\ldots010\ldots0) = \Pr(z_i = 1).$$

This is equivalent to

$$f(0\ldots0) + f(0\ldots010\ldots0) = 2\Pr(z = 1).$$

This is independent of s, the coordinate that was free, hence all f(0...010...0) are equal, independently of s. From now on we let f_1 denote the value of any f(0...010...0) and f_0 the value f(0...0).

We take another step and examine (4:2) under the assumption that $t = m - 1 = c - 2$, with two bits free, s_1 and s_2. In the same way as above, we get that (4:2) implies that:

$$f(0...010...010...0) + f(0...010...000...0) + f(0...000...010...0) + f(0...0)$$
$$= f(0...010...010...0) + 2f_1 + f_0 = 4\Pr(z_i = 1).$$

As previously, this is independent of s_1 and s_2, and we will denote all f(00...010...010...0) by f_2.

Using induction we can do this with more and more free bits, and the final result is that $f(x)$ is only dependent on the number of ones in x. This proves the lemma.

❑

We generalize the notation in the proof by the following definition.

Definition 4.4: With f_j we denote the value of $f(x)$ for any x such that the Hamming weight $w_H(x) = j$.

For example, f_3 is the value of $f(x)$ for any x with $w_H(x) = 3$ and f_{17} is the value of $f(x)$ for any x with $w_H(x) = 17$.

From expression (4:2) we remember that the definition of mth-order correlation immunity in definition 4.2 is equivalent to:

$$\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y) = \Pr(z_i = 1) \text{ for every choice of } j_1, \ldots, j_t \in [1, c] \text{ all different, and any } y \in \{0, 1\}^t, \text{ for all } t \leq m.$$

Let us examine the left-hand side of this equation.

$$\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y)$$

$$= \frac{\Pr((z_i = 1) \cap ((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y))}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y)}$$

$$= \frac{\sum_{x^{(i)} | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y} \Pr(x^{(i)}) \Pr(z_i = 1 | x^{(i)})}{\Pr((x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y)}$$

$$= \frac{\sum_{x^{(i)} | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y} 2^{-c} f(x^{(i)})}{2^{-t}} = \frac{\sum_{x^{(i)} | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y} f(x^{(i)})}{2^{c-t}}$$

This means that the definition of *m*th-order correlation immunity in definition 4.2 is equivalent to:

$f(x)$ is *m*th-order correlation-immune if

$$\frac{\sum_{x^{(i)} | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y} f(x^{(i)})}{2^{c-t}} = \Pr(z_i = 1) \text{ for all } j_1, j_2, \ldots, j_t, t \leq m$$

and all $y \in \{0, 1\}^t$. (4:4)

Lemma 4.3: If *f(x)* is a maximally correlation-immune function, then $f_k + f_{k+1} = 2\Pr(z_i = 1)$.

Proof: Assume that *f* is maximally correlation-immune. From lemma 4.2 we have that $f_k = f(x)$ for any x such that $w_H(x) = k$. Specifically, $f_k = f(0\ldots01\ldots10)$, that is, when x is a vector consisting of first $c-k-1$ zeros, followed by k ones and finally a single zero. In the same way $f_{k+1} = f(0\ldots01\ldots11)$, with x being a vector of $c-k-1$ zeros followed by $k+1$ ones.

Choosing $y = (0\ldots01\ldots1)$ with $c-k-1$ zeros and k ones, in expression (4:4) then yields:

$$\frac{f_k + f_{k+1}}{2} = \frac{f(0\ldots01\ldots10) + f(0\ldots01\ldots11)}{2}$$

$$= \frac{\sum_{x^{(i)} | (x_1^{(i)}, \ldots, x_{c-1}^{(i)}) = y} f(x^{(i)})}{2} = \Pr(z_i = 1).$$

❑

Lemma 4.4: If *f(x)* is a maximally correlation-immune function then *c* is odd and

$$f_j = \begin{cases} 0 & j \text{ even} \\ 2\Pr(z_i = 1) & j \text{ odd} \end{cases} . \qquad (4:5)$$

Proof: Assume that *f* is maximally correlation-immune. Lemma 4.3 implies that $f_j + f_{j+1} = 2\Pr(z_i = 1)$ for every *j*. We know that $f_0 = 0$, since when $x^{(i)} = \bar{0}$ the pirates cannot choose $z_i = 1$ (by the Marking Assumption). $f_0 = 0$ together with $f_j + f_{j+1} = 2\Pr(z_i = 1)$ yields $f_1 = 2\Pr(z_i = 1)$, which yields $f_2 = 0$ and so on.

Just as we know that $f_0 = 0$ we know that $f_c = 1$, which, according to (4:5), is impossible if *c* is even. Hence, *c* is odd if *f* is maximally correlation-immune.

❑

Lemma 4.5: If *f(x)* is a maximally correlation-immune function then $\Pr(z_i = 1) = 1/2$.

Proof: Assume that *f* is maximally correlation-immune. Lemma 4.4 implies that $f_j = 2\Pr(z_i = 1)$ when *j* is odd. From lemma 4.4 we also have that *c* is odd if *f* is maximally correlation-immune. Choosing $j = c$ we get that $\Pr(z_i = 1) = (f_{c-1} + f_c)/2 = (0 + 1)/2 = 1/2$.

❑

Lemmas 4.4 and 4.5 assume that the probability of ones in the creation of the fingerprinting code is neither 0 nor 1.

Theorem 4.6: f is a maximally correlation-immune if and only if $\Pr(z_i = 1) = 1/2$, c is odd and

$$f_j = \begin{cases} 0 & j \text{ even} \\ 1 & j \text{ odd} \end{cases} . \tag{4:6}$$

Proof: Lemmas 4.4 and 4.5 gives the implication in one direction. To show it in the other direction, assume that $\Pr(z_i = 1) = 1/2$, c is odd and

$$f_j = \begin{cases} 0 & j \text{ even} \\ 1 & j \text{ odd} \end{cases} .$$

Let us for a while identify the real values 0 and 1 with the binary values '0' and '1'. We see that expression (4:6) is equivalent to $z_i = f(x^{(i)}) = x_1^{(i)} \oplus x_2^{(i)} \oplus \ldots \oplus x_c^{(i)}$, where $\oplus$ means addition modulo 2. We use this in expression (4:2), setting $m = c - 1$, using any $j_1, \ldots, j_t \in [1, c]$ all different, any $y \in \{0, 1\}^t$ and any $t \leq m = c - 1$. We get:

$$\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y)$$

$$= \Pr(x_1^{(i)} \oplus x_2^{(i)} \oplus \ldots \oplus x_c^{(i)} = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y) .$$

But there is at least one $x_j^{(i)}$ among $x_1^{(i)}, \ldots, x_c^{(i)}$ not in $x_{j_1}^{(i)}, \ldots, x_{j_t}^{(i)}$. From the code construction we know that $\Pr(x_j^{(i)} = 1) = 1/2$, and that $x_j^{(i)}$ is independent of all the other bits in the input vector. Thus the random variable

$x_1^{(i)} \oplus \ldots \oplus x_c^{(i)}$ is independent of the random variable $x_{j_1}^{(i)} \oplus \ldots \oplus x_{j_t}^{(i)}$, so that $\Pr(z_i = 1 | (x_{j_1}^{(i)}, x_{j_2}^{(i)}, \ldots, x_{j_t}^{(i)}) = y) = \frac{1}{2} = \Pr(z_i = 1)$.

❑

Except for the need for c to be odd, this result is the same as that of [TS84], and it is also from there the idea came.

Using a combining function like that in (4:6), the pirate group will make it impossible to use correlation based methods to test if a set of users is a subset of the entire pirate group. On the other hand, since (4:6) is a strictly deterministic combining function, a test (not an algorithm for *finding* the collusion) performed on the whole pirate group will be easy to do and it will be very strong. If the whole pirate group is tested against the illegal fingerprint created according to (4:6), the illegal fingerprint is the modulo-2 sum of the pirates fingerprints, and the probability of this happening for an innocent group is 2^{-n}, a probability that will be extremely small for practical values of n.

4.3 Discussion

The pirates cannot distinguish between embedded ones and zeros in the object. In the case where the number of pirates is odd this does not matter, as it is the input that occurs an odd number of times that should be the output for the combining function to be maximally correlation-immune, and the pirates do not have to be able to distinguish between embedded ones and zeros to be able to do that. Additionally, as the number of pirates is odd, the fact that the pirates do not have any choice in what to output in the undetectable positions does not matter. In the undetectable positions the embedded bit value occurs an odd number of times and is therefore the correct value to output to the illegal fingerprint.

We have shown that there is exactly one combining function f that is maximally correlation-immune, and that if the number of pirates is odd it is possi-

ble for the pirates to implement that function under the restrictions that the Marking Assumption imposes.

We have also seen that, even though a maximally correlation-immune combining function prevents testing on subsets of the whole pirate group, it makes testing on the whole group both strong and easy to perform.

Chapter 5

A Testing Method

In the random code model the concept of correlation introduced in chapter 4 can be used for testing whether a proposed group P is guilty of having created a specific illegal fingerprint z. This can be done in several ways, and the method proposed here is a rather straightforward one. Our approach is that we will consider only the positions that are undetectable for the proposed group and declare the group guilty if its fingerprints agree with the illegal fingerprint in all undetectable positions, and if the probability of this happening by chance for a non-guilty group is low enough. (This can be said to rely on lemma 4.1; under the Marking Assumption, there is no combining function $f(\{0,1\}^c) \to \Re[0,1]$ that is cth order correlation-immune.)

When using an illegal fingerprint to test whether a certain group of pirates is guilty, we do a hypothesis test, and can thus make two kinds of error: failing to accuse a group that is guilty, and accusing a group that is not guilty. We want the probability of both these kinds of error to be small, which leads to a trade-off between conflicting goals. These probabilities are, of course, dependent on the fingerprint length and on the number of pirates in the collusion. We will denote the probability of failing to accuse a guilty group by p_{fa} and the probability of accusing an innocent group by p_{ai}.

5.1 Method Description

When examining a group of possible pirates, we first count the number of coordinates i in which the input vector $x^{(i)}$ (see definition 4.1) is either $\bar{0}$ or $\bar{1}$ and denote this number N. (With $\bar{0}$ and $\bar{1}$ we mean the vectors of all zeros and all ones, respectively.) The number of coordinates i in which $z_i = 0$ when $x^{(i)} = \bar{0}$ plus the number of positions in which $z_i = 1$ when $x^{(i)} = \bar{1}$ is denoted r. If the group is innocent, r will be an outcome of a random variable taken from a binomial distribution, $r \in \mathrm{Bi}(N, 2^{-1})$, where N is a binomially distributed random variable, $N \in \mathrm{Bi}(n, 2^{1-c})$, and c is the number of users in the group. The probability that r will be close to N will be small, and it will tend to 0 as N grows.

We will accuse the group of being guilty if $r = N$ and $N > N_t$, where N_t is a threshold. (N being large means that the probability of r being equal to N by chance is small.) The probability of accusing a given innocent group is thus $p_{\mathrm{ai}} = \Pr(N > N_t, r = N)$, and the probability of accusing a guilty group is $1 - p_{\mathrm{fa}} = \Pr(N > N_t)$. We can make different trade-offs between these probabilities by choosing different values of N_t.

In the method described, we only take into account the positions with input vectors equal to $\bar{0}$ or $\bar{1}$. The reason for this is that the pirates have no freedom of choice in these positions. The output has to be equal to the bits in the input vector, and hence, the output in these positions is totally deterministic (for a given group of pirates). For all other positions the pirates can choose what value to output, and can thereby possibly remove any information usable in the testing.

To simplify the problem the probability of accusing a group consisting in part of pirates and in part of innocent users is not considered here. The reason for this is that when considering groups that are partly guilty the strategy of the pirates makes a difference. For example, if a collusion of size $c \geq 3$ where c is odd, makes a majority choice in every position, then any single pirate codeword can be exchanged for any innocent codeword, and the resulting, partly guilty group, will have every undetectable position equal to the illegal word. On the other hand, a collusion using the majority choice strategy creates a correlation between the illegal word and the single pirate codewords, so they

can easily be found individually in that way. We can see that the problem becomes dependent of the pirates' strategy, and thus complicated when we examine partly guilty groups.

5.2 Analysis

When performing the testing, we can make two kinds of error, corresponding to the first and second kinds of error in hypothesis testing. In this case this means accusing a group that is not guilty and failing to accuse a group that is guilty. The probabilities of making these two kinds of error are, of course, interesting and we will find expressions for them in the following.

The method will falsely accuse an innocent group P if $r = N$ and $N > N_t$. The probability of this happening when the group is innocent is:

$$p_{\text{ai}} = \sum_{i = N_t + 1}^{n} \Pr(N = i)\Pr(r = N | N = i) = \sum_{i = N_t + 1}^{n} 2^{-i}\Pr(N = i)$$

$$= \sum_{i = N_t + 1}^{n} 2^{-i}\binom{n}{i}2^{i(1-c)}(1 - 2^{1-c})^{n-i}$$

$$= \sum_{i = N_t + 1}^{n} \binom{n}{i}2^{-ic}(1 - 2^{1-c})^{n-i} \quad . \qquad (5:1)$$

The method will fail to accuse a guilty group P if $N \leq N_t$, that is, if the number of undetectable positions for P is too small. The probability of this happening is:

$$p_{\text{fa}} = \sum_{i=0}^{N_t} \Pr(N = i) = \sum_{i=0}^{N_t} \binom{n}{i} 2^{i(1-c)} (1 - 2^{1-c})^{n-i}. \quad (5:2)$$

Both (5:1) and (5:2) are easy to compute numerically. In figure 5.1 below we see the possible trade-offs for different numbers of pirates and different fingerprint lengths. By choosing different N_t it is possible to choose the combination of error probabilities from the points on the corresponding curve. (Since N_t is an integer, $0 \leq N_t < n$, the number of possible choices on each curve is finite, but the points are quite dense. The number of points on the interesting ranges of the curves range from 250 on the leftmost, to 25 on the rightmost.)

The right end points of the curves, $N_t = 0$, correspond to the lowest possible probabilities of failing to accuse a guilty group for that specific combination of c and n. No other points exist below these.

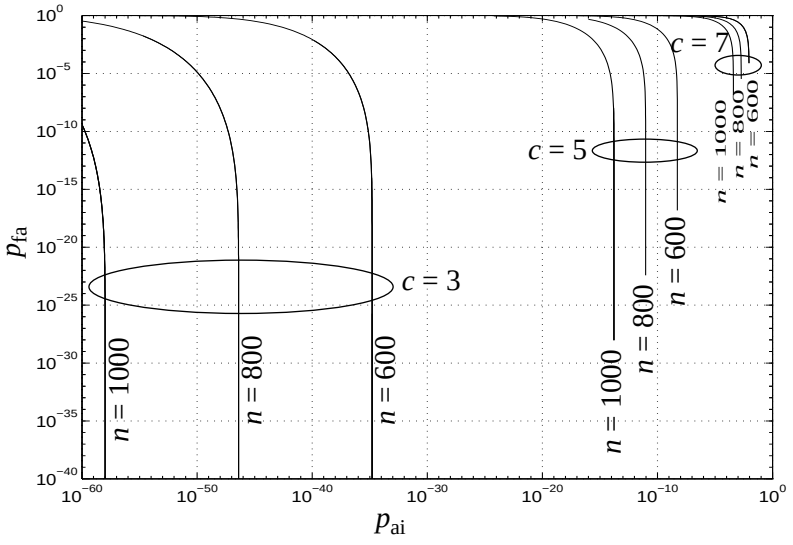

Figure 5.1: *Possible trade-offs between the probability of accusing an innocent group and probability of failing to accuse a guilty group. Different curves are for different numbers of pirates, c, and different fingerprint lengths, n. This is further described in section 5.2*

If we fix a pair of error probabilities, p^*_{fa} and p^*_{ai}, and for different values on c find the smallest n such that it is possible to choose $p_{\mathrm{fa}} \leq p^*_{\mathrm{fa}}$ and $p_{\mathrm{ai}} \leq p^*_{\mathrm{ai}}$, we can see how the fingerprint lengths needed to achieve these error probabilities vary with the number of pirates. The fingerprint lengths needed to be able to keep the two error probabilities below 10^{-4} for different numbers of pirates are shown in figure 5.2. At least in this range the fingerprint length needed grows exponentially with the number of pirates in the collusion.

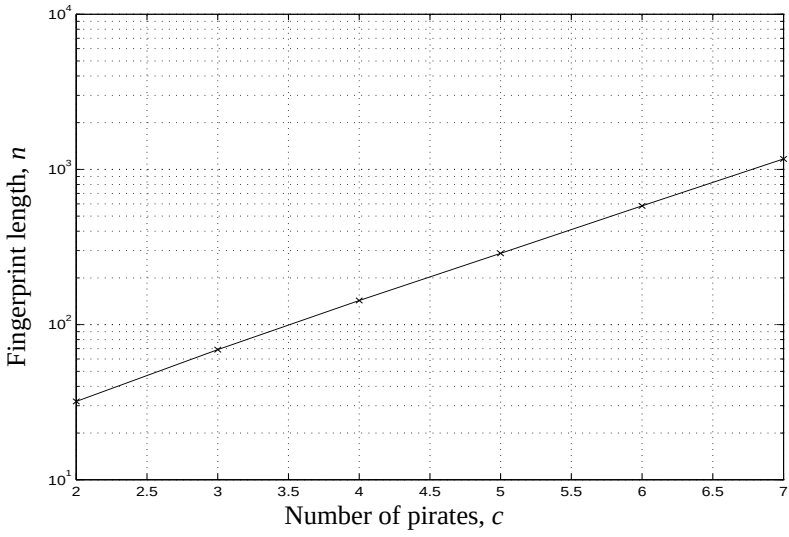

Figure 5.2: *The minimum fingerprint lengths needed to keep both error probabilities less than 10^{-4} for different sized pirate groups.*

5.3 Discussion

In figure 5.1 we see that the number of pirates has a strong influence on the performance of the testing method. On the other hand, the number of users in the system, M, does not affect the performance at all.

If we fix the error probabilities, we can see in figure 5.2 that the fingerprint length needed grows exponentially with the number of pirates, which, of course, is not satisfactory. On the other hand, in the same figure we see that we can deal with up to seven pirates with slightly more than 1000 bits in the fingerprints and error probabilities less than 10^{-4}. It is important to remember, though, that this is a *testing* method, not a method for tracing the pirates. It can only be used for testing whether or not a certain, proposed group of users should be accused of being guilty of having created a certain illegal copy. Furthermore, we have not considered the probability of accusing a group consisting in part of pirates and in part of innocent users.

Chapter 6

Pirate Strategies

When pirates create an illegal copy they do it in some way that they believe fits their purpose – they use a strategy. For example, they might choose a strategy that minimizes the probability of getting caught, or that maximizes the probability of framing another, specific user. Whatever their purpose, this raises the question of what a strategy is and what the set of strategies from which the pirates can choose looks like. These are the questions that this chapter will address, in the setting of the known code model.

The contents of this chapter were assembled in cooperation with Tina Lindkvist, also a Ph.D. student at the Department of Electrical Engineering, Linköping University.

6.1 Introduction

Knowledge of what a strategy is and what strategies there are is essential when designing a fingerprinting system. The designer wants to find an optimal fingerprinting code, and given the code find the optimal tracing method. Optimality can be defined in a min-max way, that is, an optimal decoding method is one that maximizes the minimum probability, over all possible pirate strategies, of tracing success. To be able to do min-max optimization

like this, it is essential to have a set containing all possible pirates strategies, and nothing but pirate strategies.

Knowledge about the pirate strategies may also help us understand the fingerprinting problem better. When we understand what the pirates can do, we might be able to find out what constitute good fingerprinting codes and good tracing methods.

The problem that we address in this chapter is to give a compact and useful description of the set of all possible pirate strategies in a fingerprinting system. This is done in section 6.3. In order to solve this problem we need to somewhat specify the known-code model of the binary fingerprinting system.

6.2 Notation and Definitions

We need some further notation and definitions in addition to what is introduced in the known code model.

- the set of c-sets of users is denoted W, so that $W = \{\{u_1, u_2, \ldots, u_c\} : u_i \in U, u_i \neq u_j \text{ when } i \neq j\}$
- a c-set of users will be denoted w, where $w = \{u_1, u_2, \ldots, u_c\} \in W$
- the pirate set is denoted $P = \{p_1, p_2, \ldots, p_c\}$, where $P \in W$.

Having introduced this notation we continue with the definitions needed for describing the pirate strategies.

Definition 6.1: The *public code* is an M-tuple denoted $\Psi = \langle \psi_{\alpha_1}, \psi_{\alpha_2}, \ldots, \psi_{\alpha_M} \rangle$, where $\{\alpha_1, \alpha_2, \ldots, \alpha_M\} = U$. Each element of the code is a binary vector of length n, and $\psi_i \neq \psi_j$ when $i \neq j$.

Definition 6.2: The *embedded code* is an M-tuple denoted $\Gamma = \langle \gamma_1, \gamma_2, \ldots, \gamma_M \rangle$. Each element of the code is a binary vector of length n, and $\gamma_i \neq \gamma_j$ when $i \neq j$.

Definition 6.3: [1]Let C_1 and C_2 be two binary codes of the same size and length n, both being defined as tuples of codewords. We say that C_1 is *equivalent* to C_2 if C_1 can be transformed into C_2 by the following operations:

1. Permutation of the coordinates.
2. Permutation of the codewords.
3. Addition modulo-2 with a binary vector of length n.

The relation between Ψ and Γ is that the embedded code is chosen randomly with equal probability among all codes equivalent to the public code.

The embedded code and its relation to the public code is a secret of the fingerprinting system.

Using the definitions so far we can describe how we assume the fingerprinting system works:

1. A public fingerprinting code Ψ is chosen by the distributor for usage in the fingerprinting system. Ψ is binary, of length n and publicly known. The original digital object that is to be fingerprinted is prepared by locating n marks in which it is possible to make alterations in such a way that the object as a whole is degraded only slightly or not at all. This is done in such a way that the Marking Assumption holds.
2. A code Γ is chosen randomly, with equal probability, among all codes equivalent to Ψ.
3. The numbering of the codewords is chosen so that $\Psi = \langle \psi_{\alpha_1}, \psi_{\alpha_2}, \ldots, \psi_{\alpha_M} \rangle$, $\Gamma = \langle \gamma_1, \gamma_2, \ldots, \gamma_M \rangle$ and γ_i is the coordinate-permuted and translated ψ_i for all $i \in U$.

1. Note that there are other ways of defining code equivalence that are more common than the one used here. The reason for this definition is the code definitions 6.1 and 6.2 that are also different from what is usually seen.

4. The objects that are to be distributed are fingerprinted by embedding the codewords of Γ into them.
5. The fingerprinted objects are distributed to the users so that user i gets the object fingerprinted with codeword γ_i.

This way of describing the embedded code as a code randomly chosen among the codes equivalent to the public code is a way of modelling that the users know what the fingerprinting code looks like, but that they cannot tell the difference between zeros and ones when they are embedded.

Definition 6.4: The *set of the pirates' embedded codewords* is denoted $\Gamma_P = \{\gamma_{p_1}, \gamma_{p_2}, \ldots, \gamma_{p_c}\}$, where γ_{p_i} are codewords in the embedded code Γ.

Definition 6.5: The *set of the pirates' public codewords* is denoted $\Psi_P = \{\psi_{p_1}, \psi_{p_2}, \ldots, \psi_{p_c}\}$, where ψ_{p_i} are codewords in the public code Ψ.

Definition 6.6: By Γ^c we denote the *set of c-sets of embedded codewords*, so that $\Gamma^c = \{\{\gamma_{u_1}, \gamma_{u_2}, \ldots, \gamma_{u_c}\} : w \in W\}$.

Definition 6.7: By Ψ^c we denote the *set of c-sets of public codewords*, so that $\Psi^c = \{\{\psi_{u_1}, \psi_{u_2}, \ldots, \psi_{u_c}\} : w \in W\}$.

Definition 6.8: Two c-sets of codewords, $X = \{x_1, x_2, \ldots, x_c\}$ and $Y = \{y_1, y_2, \ldots, y_c\}$, none, one or both public, are said to be *c-set equivalent* if X can be transformed into Y by permutation of the coordinates and addition modulo-2 with a binary vector of length n. This is denoted $X \equiv Y$.

There are three things worth noting here. The first is that c-set equivalence really is an equivalence relation since it is reflexive, symmetric and transitive. The second is that $\{\psi_{u_1}, \ldots, \psi_{u_c}\} \equiv \{\gamma_{u_1}, \ldots, \gamma_{u_c}\}$ for all $\{u_1, u_2, \ldots, u_c\} \in W$, which means that, regarding c-set equivalence, we do not have to distinguish between c-sets of public and embedded codewords. The third is that groups of users can, for any observed c-set of fingerprinted objects, tell which c-set equivalence class the corresponding c-set of embedded (as well as public) codewords belongs to.

Definition 6.9: With Ψ_w^c we denote the *set of equivalent c-sets of public codewords*, so that $\Psi_w^c = \{\psi^c \in \Psi^c : \psi^c \equiv \{\psi_{u_1}, \ldots, \psi_{u_c}\}\}$ where $\psi^c = \{\psi_{i_1}, \ldots, \psi_{i_c}\}$, w is the c-set $\{u_1, u_2, \ldots, u_c\} \in W$ and $\psi_{u_j} \in \Psi$ for all $j \in \{1, \ldots, c\}$.

The corresponding definition of Γ_w^c, can be made for the embedded codewords. There is a one-one correspondence between the different Ψ_w^c and the different Γ_w^c, defined by the c-set equivalence relation. Ψ_w^c corresponds to Γ_w^c if the elements in Ψ_w^c are equivalent to the elements in Γ_w^c. This means that a user group (for example the pirate group) can tell which equivalence class Ψ_w^c their c-set of embedded codewords corresponds to, even though they do not know what the different Γ_w^c look like.

Definition 6.10: With Λ we denote the *set of equivalence classes of c-sets of public codewords*, so that $\Lambda = \{\Psi_w^c : w \in W\}$.

The c-set equivalence relation defines a partitioning of Ψ^c, and the equivalence classes are the elements in Λ. By observing in which way the objects in a set of objects differ from each other, groups of users (as well as groups of pirates) can tell which equivalence class Ψ_w^c their c-set of embedded codewords corresponds to. They cannot make any classification of the equivalent c-sets of codewords since they all look the same to them.

Definition 6.11: With X^c we denote the *set of c-vectors*, so that $X^c = \mathrm{GF}(2)^c$.

Definition 6.12: Let $w \in W$ and assign a specific but arbitrary order to the elements in w. The vector consisting of the bit values of the codewords of w in coordinate i, in order, is called the ith *input vector* of w and is denoted x_i, where $x_i \in X^c$. This definition is valid both for the public and the embedded codewords.

If x is a binary vector, we denote by $\tilde{x}$ the binary complement of x.

Definition 6.13: We say that two input vectors x and y are equivalent if $x = y$ or if $x = \tilde{y}$, and we denote this $x \equiv y$.

Definition 6.14: With X_y^c we denote the *set of equivalent c-vectors*, so that $X_y^c = \{x : x \in X^c, x \equiv y\}$.

Definition 6.15: With Ξ^c we denote the *set of c-vector equivalence classes*, so that $\Xi^c = \{X_y^c : y \in X^c\}$.

As in the case of the *c*-sets, a user group can, if they assign an ordering of their objects, tell which equivalence class of *c*-vectors any detectable coordinate belongs to. Also, they cannot tell the difference between coordinates in the same equivalence class.

Definition 6.16: Let $w \in W$ with an arbitrary but specified order, and let $X_y^c \in \Xi^c$. The set of coordinates i for which the input vectors $x_i \in X_y^c$ is called the *input vector coordinate set* of type X_y^c and is denoted $C_y^w \subseteq \{1, 2, \ldots, n\}$.

Note that since for every coordinate *i*, the input vector x_i belongs to exactly one equivalence class of *c*-vectors X_y^c, the sets C_y^w form a partitioning of the set of coordinates $\{1, 2, \ldots, n\}$.

6.3 Representation of the Pirate Strategies

We want to find a compact way to represent all pirate strategies. With "strategy" we mean anything that the pirates choose to do to create their illegal fingerprint under the restrictions imposed earlier in this chapter. In practice this means that they can in each detected mark choose the alternative corresponding to '0', the alternative corresponding to '1' or something else, corresponding to an erasure, which we will denote by *'e'*. Since the pirates might want to behave differently depending on what their fingerprinted objects look like, we will demand that a strategy is defined so that they can behave differently for every different appearance of their set of fingerprints, or equivalently, for every equivalence class of *c*-sets of public codewords.

Without knowledge of which codewords the pirates have, the exact result of the pirates' strategy will in general be unknown to them. To the pirate collusion, a strategy is a random mapping from the set of equivalence classes of *c*-

sets of public codewords to a subset of all possible fingerprints, possibly with erasures, $s:\Lambda \rightarrow B \subseteq (\mathrm{GF}(2) \cup \{e\})^n$, where B is the set of words that can be created by the pirate group (the exact appearance of B is unknown to the pirates). Given the codewords of the pirate set and a numbering, a strategy is a deterministic mapping from the set of c-sets of embedded codewords to the set of all possible fingerprints, possibly with erasures, $s:\Gamma^c \rightarrow (\mathrm{GF}(2) \cup \{e\})^n$.

We assume that there is exactly one pirate collusion (if there are more collusions than one they can be treated separately) and that the collusion chooses its strategy before the fingerprinted objects are distributed. We further assume that the pirates arbitrarily number themselves from 1 to c, where c is the size of the collusion.

Theorem 6.1: Any pirate strategy, s_P, can be represented in the following way, and everything represented this way is a strategy that a pirate group can perform:

1. For every equivalence class of c-sets of public codewords, $\lambda \in \Lambda$, choose a specific strategy.
2. *Specific Strategy (for λ):* Choose an arbitrary representative $\lambda^* \in \lambda$ for the equivalence class λ, and order the codewords in λ^* arbitrarily to be able to define the input vectors of λ^*. For every c-vector equivalence class $\xi \in \Xi^c$ in λ^*, choose how many of the bits in the illegal copy should be chosen equal to γ_{p_1}, the first codeword in order in λ^*, and how many should be erased. All other bits are chosen to be different from γ_{p_1}, but not as erasures. The number of bits equal to γ_{p_1} we denote t_ξ^λ, and the number of bits to be erased we denote ε_ξ^λ.

Execution of the chosen strategy: When the fingerprinted objects have been distributed, find the $\lambda_{obs} \in \Lambda$ such that Γ_P is equivalent to the elements in λ_{obs}. Order the pirate codewords so that the sizes of the c-vector equivalence classes coincide with those of the chosen representative λ^* from the number 2 just above.

Then, in every input vector coordinate set of type ξ, with $\xi \in \Xi^c$, erase any $\varepsilon_\xi^{\lambda_{obs}}$ bits, choose any $t_\xi^{\lambda_{obs}}$ of the remaining bits equal to that of γ_{p_1}, and the rest different from γ_{p_1} but not as erasures.

Proof:

1. It is sufficient to allow different specific strategies only for different *c*-set equivalence classes. Further separation, into smaller classes, cannot be told apart by the pirates who therefore cannot apply different actions for equivalent *c*-sets of codewords. The *c*-set equivalence relation partitions the set of *c*-sets of codewords, so it is possible to describe any strategy in terms of what to do depending on which equivalence class $\lambda \in \Lambda$ the pirates' *c*-set of embedded codewords corresponds to. Since it is possible for the pirates to tell which $\lambda \in \Lambda$ their *c*-set of embedded codewords corresponds to, it is possible for them to do this separation into specific strategies.

2. Let $\Gamma_P = \{\gamma_{p_1}, \gamma_{p_2}, \ldots, \gamma_{p_c}\}$ be the pirates *c*-set of embedded codewords, with an arbitrary ordering. In all detectable coordinates, that is, coordinates $i \notin C_{\bar{0}}^P$ where $\bar{0}$ is the *c*-vector consisting only of zeros, the pirates can choose to output to the illegal fingerprint an erasure, the bit value in γ_{p_1} or the bit value not in γ_{p_1}. No other choices of what to output are possible (the choice of γ_{p_1} as reference is arbitrary; any $\gamma \in \Gamma_P$ would serve equally well). This means that there are $3^{n-\left|C_{\bar{0}}^P\right|}$ different things (words with some bits possibly erased) that the pirates can output to the illegal copy. Note that this does not depend on which ordering of the codewords was chosen.

   We let $\lambda_{obs} \in \Lambda$ be the set such that Γ_P is equivalent to the elements in λ_{obs}. This is observable to the pirates. We also impose an order on the elements in Γ_P so that the sizes of the *c*-vector equivalence classes coincide with those of the chosen representative λ^*.

   We now arbitrarily number all the input vector coordinate sets except $C_{\bar{0}}^P$ as $C_1^P, C_2^P, \ldots, C_{2^{c-1}-1}^P$. Without loss of generality we can assume that the pirates will take the coordinate sets $C_1^P, C_2^P, \ldots, C_{2^{c-1}-1}^P$ in turn, and decide what to output to the coordinates in the illegal copy (erasure, the bit value in γ_{p_1} or the bit value not in γ_{p_1}). This means that they can choose

their actions for each input vector coordinate set separately, without reducing their range of specific strategies to choose from.

It is sufficient to allow different behaviour only for different input vector coordinate sets C_y^P. Further separation, into smaller classes, cannot be told apart by the pirates, and thus they cannot behave differently depending on this.

Knowing that each input vector coordinate set can be treated separately, we turn to investigate what the pirates can do in each such specific set. Let C_y^P be any input vector coordinate set except C_0^P. In each of the $\left|C_y^P\right|$ coordinates of C_y^P the pirate collusion has to choose whether to output to the illegal copy an erasure, the bit value in γ_{p_1} or the bit value not in γ_{p_1}. Only one alternative is possible in each coordinate. Assume that the pirates choose to output an erasure in ε coordinates $\{k_\varepsilon\}$, the bit value in γ_{p_1} in t coordinates $\{k_t\}$, and the bit value not in γ_{p_1} in $\left|C_y^P\right| - t - \varepsilon$ coordinates $C_y^P \backslash (\{k_\varepsilon\} \cup \{k_t\})$. (the sets $\{k_t\}$, $\{k_\varepsilon\}$ and $C_y^P \backslash (\{k_\varepsilon\} \cup \{k_t\})$ are all disjoint).

Since the embedded code is chosen randomly among all codes equivalent to the public code, the pirates cannot tell which of the coordinates in the public code that corresponds to the ε coordinates $\{k_\varepsilon\}$ and the t coordinates $\{k_t\}$, only that they are ε and t of the $\left|C_y^P\right|$ coordinates with public input vectors equivalent to y that exist in each of the c-sets in λ_{obs} (given a correct ordering). Furthermore, since the embedded code has been chosen with equal probability among all codes equivalent to the public code, all mappings between the possible coordinates in the public code, and the coordinates in C_y^P are equally probable. This is true regardless of how the coordinate sets $\{k_t\}$, $\{k_\varepsilon\}$ and $C_y^P \backslash (\{k_\varepsilon\} \cup \{k_t\})$ are chosen by the pirates, and thus the choice of *which* coordinates is irrelevant, only the

number of coordinates for each type (erasures, bits equal to γ_{p_1} and bits not equal to γ_{p_1} but not erasures) matters.

It is possible for the pirates to execute this kind of specific strategy, regardless of how it is chosen.

All in all this means that this strategy representation is sufficiently detailed to describe any strategy that the pirates can employ, but also that the pirates can employ any strategy represented like this.

❑

This result means that we can reduce the number of specific pirate strategies to consider from $3^{n-\left|C_0^P\right|}$, taken from the proof of theorem 6.1, to

$$\prod_{i=1}^{2^{c-1}-1} \binom{\left|C_i^P\right|+2}{2}$$

(for each C_i^P, $i \in [1, 2^{c-1}-1]$, the pirates choose how many coordinates will be equal to γ_{p_1}, how many will be erasures and how many will be unequal to γ_{p_1} but not erasures).

6.4 A Strategy Example

Let the public fingerprinting code,

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

where the codeword ψ_i, $1 \le i \le 5$, is the ith row in the code matrix, and assume that the number of pirates, c, is 2. In this case the set of equivalence classes of codewords Λ consists of two elements, which can be represented by the two sets below.

$$\Psi^2_{\{1,2\}} = \left\{ (1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0), (0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0) \right\}$$

$$\Psi^2_{\{4,5\}} = \left\{ (0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0), (0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1) \right\}$$

Both $\Psi^2_{\{1,2\}}$ and $\Psi^2_{\{4,5\}}$ have all of their detectable input vectors in only one c-vector equivalence class, and it is the same such class for both $\Psi^2_{\{1,2\}}$ and $\Psi^2_{\{4,5\}}$ and regardless of the chosen order of the codewords.

Choosing the strategy for the pirates in this case consists of choosing specific strategies for the equivalence classes represented by $\Psi^2_{\{1,2\}}$ and $\Psi^2_{\{4,5\}}$. We assume that the pirates will not use erasures and find that the specific strategies can be represented by integers, between 0 and 4 for $\Psi^2_{\{1,2\}}$ and between 0 and 5 for $\Psi^2_{\{4,5\}}$, telling how many bits in each case should be chosen equal to the first codeword in the chosen order (though the order does not make any difference in this case). Thus, without erasures there are $5 \times 6 = 30$ different strategy representations in this simple fingerprinting system.

6.5 Discussion

This chapter's main result is theorem 6.1, a compact way of representing any pirate strategy in a binary fingerprinting system. Every representation, in the sense of theorem 6.1, corresponds to a strategy that the pirates can use, and every strategy that the pirates can use corresponds to such a representation.

To every strategy representation there is a subset of the whole fingerprint space such that by executing the strategy an element from this subset is randomly, but not necessarily uniformly, chosen as the illegal fingerprint that the pirates will distribute.

It is the secrets of the fingerprinting system that groups specific actions into sets of actions that cannot be told apart by the pirates, sets which we identify with the strategies and the strategy representations. This grouping of specific actions has the effect of restricting the accuracy of the pirates' construction of the illegal fingerprint, resulting in the word created by the pirates being chosen randomly from a non-empty subset of the whole fingerprint space.

As we have seen in previous chapters, the pirates' chosen strategy is essential to the outcome of the tracing or testing. Now that we have this compact representation of all pirate strategies we can, at least if the code isn't too large, analyse all the different pirate strategies and find the strongest one.

Chapter 7

Fingerprinting and Game Theory

In this chapter we give a game theoretical description of the fingerprinting problem. This is interesting since knowing that fingerprinting can be described in game theoretical terms gives us new tools to explore and understand the problem. In principle, given a specific code, it will be possible to find the optimal behaviour both for the collusion and for the distributor, and by comparing results for different codes it will be possible to find optimal codes for given code parameters.

The contents of this chapter were assembled in cooperation with Tina Lindkvist, also a Ph.D. student at the Department of Electrical Engineering, Linköping University.

7.1 Introduction

A fingerprinting scheme like the one we are considering can be interpreted as a game played between the pirate collusion and the distributor. The goal of the collusion is to create an illegal copy such that the distributor will not accuse any of the participating pirates when the tracing method is applied to it. In the game we will model this by letting the different tracing methods correspond to the rows and the different pirate strategies to the columns of the

payoff matrix of the game. A successful tracing will be assigned the value 1 and a failed tracing the value 0 (both values being real). Hence, in the game the distributor will want to maximize the payoff and the collusion will want to minimize it.

We will derive the results in the case that the pirates cannot make erasures. The generalizations necessary to allow for erasures are totally straight-forward.

7.2 Game Theory

In this section we will present the parts of the theory of games that we will need in the rest of the chapter. For a more thorough description, see for example [NNV77].

7.2.1 Matrix games

We will use the concept of *matrix games* from the theory of games. In a game like this there are two players, both wanting to achieve the greatest possible individual gain. The game is such that what one player wins, the other loses and vice versa.

The players are numbered I and II, each with a finite set S_i of strategies to choose from. A *situation* $s = (s_I, s_{II})$ is the combined, by the players individually chosen, strategies for a round in the game. There is also a payoff function $H(s)$ that describes the gain of player I, which is equal to what player II loses. This kind of game is called a zero-sum game.

Any matrix game can be described with a matrix H, with the rows corresponding to the strategies of player I and the columns corresponding to the strategies of player II. The element in row i and column j is what player I gains and what player II loses, when player I has chosen strategy i and player II strategy j. We will refer to this element as $H(i, j)$.

7.2.2 The Choice of Strategies

A reasonable way for player I of playing the game is to choose a strategy (row in the matrix) such that he gets as large a payoff as possible when player II does his best to minimize the payoff. The payoff for player I when choosing strategy i if player II is minimizing the payoff is $\min_j H(i, j)$, so when maximizing this player I will be sure to get at least $\max_i \min_j H(i, j)$.

It is possible to follow the same line of reasoning for player II as well, the only difference being that he will minimize the largest payoff that player I can choose, that is, minimize $\max_i H(i, j)$, so that he will be sure that the payoff will be no more than $\min_j \max_i H(i, j)$.

Lemma 7.1: $\max_i \min_j H(i, j) \leq \min_j \max_i H(i, j)$

For a proof of lemma 7.1, see, for example [NNV77].

When the left-hand side and right-hand side in lemma 7.1 are equal, we call this the *value* of the game. When a game has a value, this is what the players will gain/lose respectively when playing the game optimally. Unfortunately not all games of this type have a value, since the left-hand side and right-hand side in lemma 7.1 may be unequal. Fortunately it is possible to extend the sets of strategies in such a way that *all* games of this kind have a value. A tuple of the players' strategies, such that the payoff is equal to the value of the game, is called a *solution* to the game.

7.2.3 Mixed Strategies

The strategies we have described so far, consisting of choosing a row (player I) or a column (player II) in the payoff matrix, are called *pure strategies*, in order to distinguish them from *mixed strategies* that will be introduced now. We extend the sets of player strategies to include probability-weighted combinations of the different pure strategies. This means that we can represent the mixed strategies with probability vectors that describe with what probability the different pure strategies are chosen. Since mixed strategies are random, the payoff of the game will be taken to be the expected pay-offs, where the expectation is taken over the probabilities of the pure strategies of the two

pirates. With mixed strategies all matrix games have a value (for a proof of this, see for example [NNV77]).

7.3 Game Theory in Fingerprinting

The game theoretical analysis of the fingerprinting problem will not be quite as clean as section 7.1 hinted at. As we will see, there are some random elements in the game, and as a result of this, we will for every pirate strategy and tracing method have to calculate the expected payoff and let that be what the game is played for. That is, the payoff function will contain the expected value of the actual payoff. This expected value is equivalent to the probability of the distributor performing a successful tracing.

7.3.1 Notation

In the rest of the chapter we will use the following notation: The distributor is denoted *D*, the pirate collusion is *P*, the set of users in the system $U = \{1, 2, \ldots, M\}$, the set of all *c*-sets of embedded codewords is Γ^c and the set of the pirates' embedded codewords is Γ_P.

7.3.2 The Distributor's Strategies

The set of distributor strategies is the set of tracing methods. We will, for now, define the set of tracing methods as the set S_D of functions $s_D: \mathrm{GF}(2)^n \rightarrow U$ from the set of all binary vectors of size *n* to the set of users in the system. This is the distributor's set of pure strategies. To ensure a solution to the game we will allow the distributor to use mixed strategies.

7.3.3 Extended Distributor Strategies

A possible generalization of the distributor's strategies is to allow what we call *memoryless random decoding* (MRD). An MRD is a mapping from $\mathrm{GF}(2)^n$ to the set of probability vectors of size $|U|$, such that the elements in the probability vector are the probabilities of decoding the illegal word to the

different users. The sum of the elements in any single probability vector must be equal to 1.

In appendix A it is shown that mixed strategies, like the ones introduced in section 7.2.3, are just as general as MRD:s, so usage of MRD:s instead of mixed strategies will not change the situation for the distributor, and we will thus not consider MRD:s.

7.3.4 The Pirates' Strategies

The pirates' strategies are a little more complex to describe. It is shown in chapter 6 how all pirate strategies in a binary fingerprinting system can be described. The only problem is that the outcome of a certain pirate strategy is in general not a specific binary word, but a word chosen randomly from a set of possible words. This means that we cannot tell whether the tracing will be successful or not – it depends on the outcome of the random choice. We will deal with this by calculating the expected payoff for each pirate strategy in the sense of chapter 6.

We will denote the set of pirate strategies by S_P and note that each element in this set is a random mapping $s_P{:}\Lambda \to \mathrm{FS}(\Gamma_P) \subseteq \mathrm{GF}(2)^n$ from the set of equivalence classes of *c*-sets of codewords (see definition 6.10) to a subset of the fingerprint space.

7.4 Putting it Together

We will now start over and describe how the payoff matrix, and thereby the game, is constructed. We want the payoff matrix $H(s_D, s_P)$ for the game between the distributor and a pirate collusion. What we can easily construct is the payoff matrix $H_{\Gamma_p}(s_D, x)$ for a game between the distributor and a specific, by the collusion Γ_P chosen, codeword x. Each element in the payoff matrix $H_{\Gamma_p}(s_D, x)$ is either 1 or 0 (real values), corresponding to a tracing success or a tracing failure, depending on whether or not s_D maps x to a codeword $\gamma \in \Gamma_P$.

We have that:

$$H(s_D, s_P) = \sum_{\Gamma_P \in \Gamma^c} \Pr(\Gamma_P) \mathrm{E}[H_{\Gamma_P}(s_D, s_P)]$$

$$= \sum_{\Gamma_P \in \Gamma^c} \Pr(\Gamma_P) \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s_D, x)$$

$$= \frac{\sum_{\Gamma_P \in \Gamma^c} \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s_D, x)}{|\Gamma^c|}. \quad (7:1)$$

Defining the payoff matrix we have shown that the game between the distributor and an arbitrary collusion is a well-defined matrix game.

7.5 A Game Theory Example

We present an example, similar to that in section 6.4. As we did there, we choose the number of pirates $c = 2$, but this time we assume the embedded code to be

$$\Gamma = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

where the codewords are the rows in the code matrix.

The set of equivalence classes of codewords Λ consists of only one element, which can be represented by the set

$$\Gamma^2_{\{1,2\}} = \left\{ (\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,), (\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,) \right\}.$$

The pirate strategies (when no erasures are used) can be represented by an integer between 0 and 4, telling how many bits should be chosen equal to the first codeword in the chosen order (though the order does not make any difference in this case). Thus, without erasures there are only 5 different strategy representations in this simple fingerprinting system.

In each coordinate only one user has a non-zero element, so if a '1' is found in any coordinate the distributor can find one of the pirates without any probability of error. This is true even if we allow erasures, since each erasure points out a pirate in the same way as a '1' does. This means that the only way for a collusion not to be traceable is to create the all-zero word. This is only possible using the pirate strategy of choosing two coordinates from the first pirate, and two from the second – all other strategies lead to the distributor being able to trace at least one pirate.

From (7:1) we know that:

$$H(s_D, s_P) = \frac{\sum_{\Gamma_P \in \Gamma^c} \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s_D, x)}{|\Gamma^c|}.$$

We note that the code is symmetric, in the sense that every collusion has the same specific strategies to choose from and in that they must construct the same word to avoid being traced. In this case it is reasonable to choose a distributor strategy that treats every collusion in the same way, that is, we can disregard exactly which user set is the collusion and choose one, for example

$$\Gamma_P = \left\{ (\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,), (\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,) \right\},$$

and without loss of generality calculate the value of the game in this case. We then get:

$$H(s_D, s_P) = \frac{\sum_{\Gamma_P \in \Gamma^c} \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s_D, x)}{|\Gamma^c|}$$

$$= \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s_D, x)\,. \tag{7:2}$$

Assume that we use the distributor strategy s, such that if the illegal word is not the all-zero word, then we accuse the user with a '1' in the coordinate that is the first non-zero coordinate in the illegal word. This strategy never accuses an innocent user, and fails to accuse a pirate only when the collusion succeeds in creating the all-zero word. In our game this means that $H_{\Gamma_P}(s, x) = 0$ if x is all-zero, and $H_{\Gamma_P}(s, x) = 1$ otherwise. Using this strategy s, we get from (7:2) that:

$$H(s, s_P) = \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s, x)\,. \tag{7:3}$$

Choosing (1100000000) as the first codeword in Γ_P we can examine (7:3) for the different pirate strategies, which we denote 0, 1, 2, 3, 4, meaning the number of coordinates to choose from the first codeword. In order to save space we do not consider coordinates 5 to 10 below, so we write $\Gamma_P = \{1100, 0011\}$ instead of $\Gamma_P = \{1100000000, 00110000\}$.

$$H(s, 0) = \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s, x)$$

$$= \Pr(0011|s_P, \Gamma_P) H_{\Gamma_P}(s, 0011) = 1 .$$

$$H(s, 1) = \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s, x)$$

$$= \Pr(1011|s_P, \Gamma_P) H_{\Gamma_P}(s, 1011) + \Pr(0111|s_P, \Gamma_P) H_{\Gamma_P}(s, 0111) +$$

$$+ \Pr(0001|s_P, \Gamma_P) H_{\Gamma_P}(s, 0001) + \Pr(0010|s_P, \Gamma_P) H_{\Gamma_P}(s, 0010)$$

$$= \frac{1}{4} \times 1 + \frac{1}{4} \times 1 + \frac{1}{4} \times 1 + \frac{1}{4} \times 1 = 1 .$$

$$H(s, 2) = \sum_{x \in \mathrm{FS}(\Gamma_P)} \Pr(x|s_P, \Gamma_P) H_{\Gamma_P}(s, x)$$

$$= \Pr(1111|s_P, \Gamma_P) H_{\Gamma_P}(s, 1111) + \Pr(1001|s_P, \Gamma_P) H_{\Gamma_P}(s, 1001) +$$

$$+ \Pr(1010|s_P, \Gamma_P) H_{\Gamma_P}(s, 1010) + \Pr(0101|s_P, \Gamma_P) H_{\Gamma_P}(s, 0101) +$$

$$+ \Pr(0110|s_P, \Gamma_P) H_{\Gamma_P}(s, 0110) + \Pr(0000|s_P, \Gamma_P) H_{\Gamma_P}(s, 0000)$$

$$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} \times 0 = \frac{5}{6} . \qquad (7:4)$$

In the same way we get $H(s, 3) = 1$ and $H(s, 4) = 1$. Thus, given the distributor strategy s, the pirates optimal strategy is the strategy "2", meaning that two bits should be chosen from each of the pirates codewords. This is the only strategy not always yielding a traceable word, and the probability of pirate success is $1 - H(s, 2) = 1/6$.

Note that this distributor strategy is not optimal in the game theoretic sense. For example, a strategy similar to s, with the only difference being that we would accuse user 1 if the all-zero word was found, would have a greater probability of correctly tracing a pirate. This is so since we assume that every collusion is equally probable, and in this case this means that if the all-zero word is found we will have the probability $1/3$ of being correct if we accused user 1. The drawback of this distributor strategy is that we would sometimes accuse an innocent user. The strategy s described above has the pleasing property of never doing this.

7.6 Discussion

Using the payoff function H from expression (7:1) we have defined a matrix game between the distributor and a pirate collusion. This game is a fully general matrix game and not some special case that is easier to analyse. Results from game theory (see section 7.2.3) state that there exists a solution in mixed strategies to every matrix game, and thus to the game defined by H. Since the game H has a solution there exist optimal strategies for both the distributor and the pirate collusion (in mixed strategies), and there is a well-defined value (probability of tracing success) for the game. Further, since there is only a finite number of codes of the same size and length, there exists for each choice of code parameters an optimal code, in the sense that there is no other code of the same size and length that yields a greater value for the game.

Knowing that fingerprinting is a game theoretical problem we know that in order to fully analyse or understand it we *have* to resort to game theory. Unfortunately game theory is often problematic to use since the problems can easily become too complex to actually analyse. This seems to be the case also

for fingerprinting (for example, the total number of pure distributor strategies in section 7.5 is $5^{(2^{10})} > 10^{715}$). Thus, to make analysis feasible we either have to examine some special case, for example codes with special properties or certain tracing methods, or we have to simplify the problem in some way, for example by studying small problem sizes or using simplified performance measures.

Chapter 8

A Performance Measure

8.1 Introduction

This chapter is somewhat disconnected from the rest of the thesis. Here we disregard the choices of the pirates as well as those of the distributor, and look only at the properties of the code. In order to do this we will introduce a performance measure with which we can evaluate codes and find good, or even optimal, codes and code constructions.

8.1.1 Terminology

Let Γ be a binary code, defined as a set, of size M and length n, and let $\boldsymbol{P}_c$ be the set of all sets of at most size c of codewords in Γ, $\boldsymbol{P}_c = \{P : P \subseteq \Gamma \mid 1 \leq |P| \leq c\}$. For any subset $X = \{x^{(1)}, \ldots, x^{(t)}\}$ of Γ of t codewords let $\xi_i = \{x_i^{(1)}, \ldots, x_i^{(t)}\}$ be the set of their values in the ith coordinate (note that $\xi_i \in \{\{0\}, \{1\}, \{0, 1\}\}$). The feasible set, $\mathrm{FS}(X)$, is then the set of words $y = (y_1, \ldots, y_n)$ such that $y_i \in \xi_i$ (this is equivalent to the definition in previous chapters). We say that a word x has the *c-identifiable parent property* (is c-i.p.p.) if $x \in \mathrm{FS}(P)$ for at least one $P \in \boldsymbol{P}_c$, and if

$$\bigcap_{P \in \boldsymbol{P}_c : x \in \mathrm{FS}(P)} P \neq \varnothing . \tag{8:1}$$

The set of c-i.p.p. words x is denoted c-IPP(Γ). If the property (8:1) holds for all x such that $x \in \mathrm{FS}(P)$ for at least one $P \in \boldsymbol{P}_c$, then the code Γ is said to be c-i.p.p.

8.1.2 Background

As in previous chapters users working together are assumed to be unable to make any changes in the objects, other than in places where their objects differ (the Marking Assumption). They can construct a new object embedded with a word from the feasible set of their codewords, and here we do not allow the pirates to make erasures. If the code is c-i.p.p. and if there are at most c users working together, any such newly constructed object x is possible to trace to at least one of the users taking part in constructing it (since there is at least one codeword that is a member of all P such that $x \in \mathrm{FS}(P)$). c-i.p.p. codes would thus solve the fingerprinting problem in this setting, so there is good reason to try to find such codes. However, to be able to construct c-i.p.p. codes the alphabet size, q, must be strictly greater than the maximum number of pirates, c (see for example [BCEKZ00]). This might not always be possible, so there is a reason for examining the fingerprinting problem also when $q \leq c$. In this chapter we look at the simplest of such cases, $q = c = 2$.

When examining i.p.p.-properties of binary codes it is reasonable not to use $c > 2$, since three pirates working together can by making a majority choice in each detectable coordinate create a word that any pair of them could have created, thus creating a word that is not c-i.p.p. If we allow erasures the pirates can do so also if the code is non-binary. This was described by Boneh and Shaw in [BS98].

8.1.3 Related Work

This chapter examines codes that do not actually have the identifiable parent property. There are however papers discussing (non-binary) codes with this property, and other related properties, e.g. frameproof codes, secure frameproof codes and traceability codes. In [SSW00] Staddon, Stinson and Wei study the relationships between the various properties and provide some

bounds and constructions. In [SvTW00] Stinson, van Trung and Wei introduce and discuss secure frameproof codes, which are related to pairwise column separable codes (PCS codes, see definition 8.2). In [SW98] Stinson and Wei investigate combinatorial properties and constructions of binary frameproof codes and traceability schemes. In [HvLLT98] Hollman, van Lint, Linnartz and Tolhuizen study upper bounds on the size of codes with the identifiable parent property. Barg, Cohen, Encheva, Kabatiansky and Zémor show in [BCEKZ00] that if the number of codewords used to create the word in the feasible set is strictly less than the size of the alphabet over which the code is defined, then in this case there exist sequences of i.p.p.-codes with asymptotically non-vanishing rates.

8.1.4 A Brief Look at the Results

We know that if the alphabet size $q \leq c$ there are no c-i.p.p. codes, so there are no binary c-i.p.p. codes if the number of pirates $c \geq 2$. Since this means that there are no binary c-i.p.p. codes of interesting sizes we would like to be able to look for codes that are "as good as possible" without being c-i.p.p. To do this we must be able to evaluate codes and see how good they are. A natural choice of performance measure would be to use some kind of error probability, but as we have argued, for example in section 7.6, it is difficult to calculate such probabilities in the general case. Instead we choose to use a combinatorial measure, dependent only on the code, defined in this way:

$$\mu_c(\Gamma) = \frac{1}{|\boldsymbol{P}_c|} \sum_{P \in \boldsymbol{P}_c} \frac{|\mathrm{FS}(P) \cap c\text{-IPP}(\Gamma)|}{|\mathrm{FS}(P)|}. \tag{8:2}$$

$\mu_c(\Gamma)$ is the average ratio of c-i.p.p. words, where the average is taken over the elements in $\boldsymbol{P}_c$, and can thus in some sense be said to be a measure of how close Γ is to being c-i.p.p.

To simplify the problem we choose to regard only the case that $c = 2$ and mainly consider a certain code class (PCS codes, see definition 8.2). Under these restrictions we will in section 8.3 show that $\mu_2(\Gamma)$ can be expressed as

$$\mu_2(\Gamma) = 1 - \frac{|\Gamma| - 2}{|\boldsymbol{P}_2|} \sum_{P \in \boldsymbol{P}_2 : |P| = 2} \frac{1}{|\mathrm{FS}(P)|}, \tag{8:3}$$

which is much easier to use for actual calculations and computations. In section 8.4 we use this expression to evaluate the performance, in the sense of $\mu_2(\Gamma)$, of several code classes defined there. Upper and lower bounds are derived in section 8.6, and in section 8.5 results of computer searches for codes with good performance are presented. Some of these searches are full searches over all code equivalence classes of a specific length and size. For slightly greater values on length and size this has not been possible, and the results for the best codes found are presented, even though it is not known whether they are optimal or not.

8.1.5 Codes

With a code we mean a subset of size M of the set of binary vectors of length n. A *code matrix* of a code is a matrix containing the codewords of the code as rows. Since all code properties we will use are the same for translated codes, we will sometimes assume, without loss of generality, that the code contains the zero word, and that this word is the first in the code matrix.

We will often describe codes by describing the columns in the code matrix. We will say that a column $(x_1, x_2, \ldots, x_M)^T$ in the code matrix is of *type j*, where $j = \sum_i x_i 2^{M-i}$. We will not use columns with only zeroes or only ones since such columns do not affect the feasible set for any subset of codewords, and hence do not affect $\mu_c(\Gamma)$.

A coordinate that differs between the codewords in a set of codewords is said to be *detectable* to that codeword set. We will also use the expression detectable in the same sense for columns and column types.

8.2 The Performance Measure

As said before, there are no binary c-i.p.p. codes of interesting sizes. To see how close to c-i.p.p. a code is, we propose to use the measure

$$\mu_c(\Gamma) = \frac{1}{|\boldsymbol{P}_c|} \sum_{P \in \boldsymbol{P}_c} \frac{|\mathrm{FS}(P) \cap c\text{-IPP}(\Gamma)|}{|\mathrm{FS}(P)|}, \tag{8:4}$$

where c-IPP(Γ) is the set of words that are c-i.p.p. More formally we make the following definition.

Definition 8.1: $c\text{-IPP}(\Gamma) = \{x : \exists \gamma \in \Gamma : \forall P \in \boldsymbol{P}_c : x \in \mathrm{FS}(P) \rightarrow \gamma \in P\} \cap$

$$\cap \{x : \exists P \in \boldsymbol{P}_c : x \in \mathrm{FS}(P)\}.$$

$\mu_c(\Gamma)$ yields a real value in $[0, 1]$, since the nominator in the sum is the size of a subset of FS(P), the denominator is the size of FS(P) and these quotients are then averaged over the elements in $\boldsymbol{P}_c$. $\mu_c(\Gamma)$ is the average ratio of c-i.p.p. words, and can thus be interpreted as a measure of how close Γ is to being c-i.p.p. For a (non-binary) code Γ that is c-i.p.p., and only then, we get $\mu_c(\Gamma) = 1$, and for a code Γ where there are no words x that are c-i.p.p., and only then, we get $\mu_c(\Gamma) = 0$.

In order to make equations smaller we will use the notation $\tilde{X}$ for the feasible set FS(X). This notation is used in many places in this chapter.

In order to work with $\mu_c(\Gamma)$ we need an alternative way of expressing (8:4).

Lemma 8.1:

$$\mu_c(\Gamma) = \frac{1}{|\boldsymbol{P}_c|}\sum_{P \in \boldsymbol{P}_c} \frac{\left|\bigcup_{x \in P}\left(\mathrm{FS}(P)\setminus \bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \mathrm{FS}(Q)\right)\right|}{|\mathrm{FS}(P)|} \tag{8:5}$$

Proof: In order to prove lemma 8.1 it suffices to show that

$$\mathrm{FS}(P) \cap c\text{-IPP}(\Gamma) = \bigcup_{x \in P}\left(\mathrm{FS}(P)\setminus \bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \mathrm{FS}(Q)\right). \tag{8:6}$$

The right side, R, can be rewritten as $R = \bigcup_{x \in P}(\tilde{P} \cap (\bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \tilde{Q})^*) = \tilde{P} \cap (\bigcup_{x \in P}(\bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \tilde{Q})^*)$, where X^* is the complement of X. We continue by studying the right part of the last expression.

$$\bigcup_{x \in P}(\bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \tilde{Q})^*$$

$$= \left\{z: \exists x \in P: z \in (\bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \tilde{Q})^*\right\}$$

$$= \left\{z: \exists x \in P: z \notin \bigcup_{Q \in \boldsymbol{P}_c: x \notin Q} \tilde{Q}\right\}$$

$$= \{z: \exists x \in P: \forall Q \in \boldsymbol{P}_c: x \notin Q \rightarrow z \notin \tilde{Q}\}$$

$$= \{z: \exists x \in P: \forall Q \in \boldsymbol{P}_c: z \in \tilde{Q} \rightarrow x \in Q\}$$

This yields the following expression for R:

$$R = \tilde{P} \cap \{z{:}\exists x \in P{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow x \in Q\}\,. \tag{8:7}$$

Using definition 8.1, and renaming some variables, we can write the left side, L, of (8:6) as

$$L = \tilde{P} \cap \{z{:}\exists \gamma \in \Gamma{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow \gamma \in Q\}\,, \tag{8:8}$$

where we have disregarded the second term in the intersection in definition 8.1, since $\tilde{P}$ in (8:8) is already a subset of this term.

We will now show (8:6) by showing inclusion in both directions. We start with $L \subseteq R$. Choose any $a \in L$. We must show that $a \in \tilde{P}$ and that $a \in \{z{:}\exists x \in P{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow x \in Q\}$. That $a \in \tilde{P}$ follows directly from the definition of L. By the definition of L we also know that $\exists \gamma^a \in \Gamma{:}\forall Q \in \boldsymbol{P}_c{:}a \in \tilde{Q} \rightarrow \gamma^a \in Q$. Since $a \in \tilde{P}$ this implies that $\gamma^a \in P$. Using that $\gamma^a \in P$ we get that $a \in \{z{:}\exists \gamma \in \Gamma{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow \gamma \in Q\}$ implies (using $x = \gamma^a$), that $a \in \{z{:}\exists x \in P{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow x \in Q\}$, which is what we want to prove.

We go on to show that $R \subseteq L$. Choose any $a \in R$. We must show that $a \in \tilde{P}$ and that $a \in \{z{:}\exists \gamma \in \Gamma{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow \gamma \in Q\}$. That $a \in \tilde{P}$ follows directly from the definition of R. By the definition of R we also know that $a \in \{z{:}\exists x \in P{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow x \in Q\}$. But $P \subseteq \Gamma$ for all P, so this implies that $a \in \{z{:}\exists \gamma \in \Gamma{:}\forall Q \in \boldsymbol{P}_c{:}z \in \tilde{Q} \rightarrow \gamma \in Q\}$, and the inclusion is proved.

❑

8.3 PCS Codes and Two Pirates

In the rest of this chapter we will consider the special case $c = 2$, and will leave out the c and write only $\boldsymbol{P}$, $\mu(\Gamma)$, IPP(Γ) and i.p.p., meaning $\boldsymbol{P}_2$, $\mu_2(\Gamma)$, 2-IPP(Γ) and 2-i.p.p. We will also, unless otherwise noted, only consider PCS codes (defined below).

Definition 8.2: The set of *pairwise column separable codes* is denoted PCSC, and consists of all codes Γ for which $|\Gamma| \geq 4$, and such that for all $A, B \subseteq \Gamma$ for which $|A| = |B| = 2$, $A = \{a, \alpha\}$, $B = \{b, \beta\}$ and $A \cap B = \varnothing$, there exists a coordinate i such that $a_i = \alpha_i \neq b_i = \beta_i$.

PCS codes are equivalent to 2-secure frameproof codes, as defined in [SvTW00].

We will analyse the performance measure $\mu(\Gamma)$ in order to find a simpler way of expressing it. According to lemma 8.1

$$\mu(\Gamma) = \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}} \frac{\left| \bigcup_{x \in P} (\mathrm{FS}(P) \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \mathrm{FS}(Q)) \right|}{|\mathrm{FS}(P)|}. \tag{8:9}$$

We will study the numerator, N, in the sum in (8:9) for an arbitrarily chosen $P \in \boldsymbol{P}$. We have that $c = 2$ and we will have to treat the two cases $|P| = 1$ and $|P| = 2$ separately. In the latter case we denote the elements in P by $p^{(1)}$ and $p^{(2)}$, so that $P = \{p^{(1)}, p^{(2)}\}$, and in the former we let $P = \{p\}$.

We will start with the case $P = \{p^{(1)}, p^{(2)}\}$:

$$N = \left| \bigcup_{x \in P} (\tilde{P} \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \tilde{Q}) \right|$$

$$= \left| \left[\tilde{P} \setminus \bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q} \tilde{Q} \right] \cup \left[\tilde{P} \setminus \bigcup_{Q \in \boldsymbol{P}: p^{(2)} \notin Q} \tilde{Q} \right] \right|$$

We will use the following notation: $\Phi = \bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q} \tilde{Q}$ and $\Theta = \bigcup_{Q \in \boldsymbol{P}: p^{(2)} \notin Q} \tilde{Q}$. We also use X^* for the complement of X.

$$N = \left|(\tilde{P} \backslash \Phi) \cup (\tilde{P} \backslash \Theta)\right| = \left|(\tilde{P} \cap \Phi^*) \cup (\tilde{P} \cap \Theta^*)\right|$$

$$= \left|\tilde{P} \cap (\Phi^* \cup \Theta^*)\right| = \left|\tilde{P} \cap (\Phi \cap \Theta)^*\right| = \left|\tilde{P} \backslash (\Phi \cap \Theta)\right|$$

$$= \left|\tilde{P}\right| - \left|\tilde{P} \cap \Phi \cap \Theta\right| \tag{8:10}$$

We introduce some further notation: $\phi = \bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \in Q} \tilde{Q}$, $\theta = \bigcup_{Q \in \boldsymbol{P}: p^{(2)} \notin Q,\, p^{(1)} \in Q} \tilde{Q}$ and $\xi = \bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \notin Q} \tilde{Q}$. Using these we can express Φ and Θ as $\Phi = \phi \cup \xi$ and $\Theta = \theta \cup \xi$. Inserting this in (8:10) we get the following:

$$\left|\tilde{P}\right| - \left|\tilde{P} \cap \Phi \cap \Theta\right| = \left|\tilde{P}\right| - \left|\tilde{P} \cap (\phi \cup \xi) \cap (\theta \cup \xi)\right|$$

$$= \left|\tilde{P}\right| - \left|\tilde{P} \cap (\xi \cup (\phi \cap \theta))\right|$$

$$= \left|\tilde{P}\right| - \left|(\tilde{P} \cap \xi) \cup (\tilde{P} \cap \phi \cap \theta)\right|. \tag{8:11}$$

To continue we need the following lemma.

Lemma 8.2: If Γ is in PCSC, $A, B \in \boldsymbol{P}$ and $A \cap B = \varnothing$ then $\tilde{A} \cap \tilde{B} = \varnothing$.

Proof: We have different cases depending on the sizes of the sets A and B. We begin with the case that $|A| = |B| = 2$ and denote the elements in A by $a^{(1)}$, $a^{(2)}$, and the elements in B by $b^{(1)}$, $b^{(2)}$. That $A \cap B = \varnothing$ means that none of $a^{(1)}, a^{(2)}, b^{(1)}, b^{(2)}$ are the same. Since Γ is a PCS code, there is at least one coordinate i such that $a_i^{(1)} = a_i^{(2)} \neq b_i^{(1)} = b_i^{(2)}$. This means that in this

coordinate every element in $\tilde{A}$ is different from every element in $\tilde{B}$, and thus that $\tilde{A} \cap \tilde{B} = \emptyset$.

Next look at the case when one of $|A|$ and $|B|$ is 1 and the other is 2. Without loss of generality we assume that $|A| = 2$ and $|B| = 1$ and denote the elements in A by $a^{(1)}$, $a^{(2)}$, and the element in B by $b^{(1)}$. We have that $\tilde{A} \cap \tilde{B} = \mathrm{FS}(A) \cap \mathrm{FS}(\{b^{(1)}\}) \subseteq \tilde{A} \cap \mathrm{FS}(\{b^{(1)}, b^{(2)}\})$, where $b^{(2)}$ is chosen so that $b^{(1)} \neq b^{(2)}$ and $A \cap B = \emptyset$. Now we can fall back on the first case and see that $\tilde{A} \cap \tilde{B} = \emptyset$.

In the case when $|A| = |B| = 1$, with $A = \{a\}$, $B = \{b\}$, we have $\tilde{A} = \{a\}$ and $\tilde{B} = \{b\}$. $A \cap B = \emptyset$ implies $a \neq b$, so we get $\tilde{A} \cap \tilde{B} = \{a\} \cap \{b\} = \emptyset$.

❑

The definition of ξ is equivalent to $\xi = \bigcup_{Q \in \boldsymbol{P}: P \cap Q = \emptyset} \tilde{Q}$ so by lemma 8.2 $\tilde{P} \cap \xi = \emptyset$. Using this in (8:11) we get:

$$N = |\tilde{P}| - |(\tilde{P} \cap \xi) \cup (\tilde{P} \cap \phi \cap \theta)| = |\tilde{P}| - |\tilde{P} \cap \phi \cap \theta|$$

$$= |\tilde{P}| - \left| \tilde{P} \cap \left(\bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \in Q} \tilde{Q} \right) \cap \left(\bigcup_{Q \in \boldsymbol{P}: p^{(2)} \notin Q,\, p^{(1)} \in Q} \tilde{Q} \right) \right|$$

$$= |\tilde{P}|$$

$$- \left| \left(\bigcup_{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \in Q} (\tilde{P} \cap \tilde{Q}) \right) \cap \left(\bigcup_{W \in \boldsymbol{P}: p^{(2)} \notin W,\, p^{(1)} \in W} (\tilde{P} \cap \tilde{W}) \right) \right| =$$

$$= \left|\tilde{P}\right| - \left| \bigcup_{\substack{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \in Q \\ W \in \boldsymbol{P}: p^{(2)} \notin W,\, p^{(1)} \in W}} \tilde{P} \cap \tilde{Q} \cap \tilde{W} \right|. \tag{8:12}$$

Choose $Q \in \boldsymbol{P}$ such that $p^{(1)} \notin Q, p^{(2)} \in Q$ and $W \in \boldsymbol{P}$ such that $p^{(2)} \notin W, p^{(1)} \in W$, and denote the elements in Q and W by q, q' and w, w' respectively. Without loss of generality we can assume that $q' = p^{(2)}$ and $w' = p^{(1)}$ so that $Q = \{q, p^{(2)}\}$ and $W = \{w, p^{(1)}\}$. If $w \neq q$ we get that $W \cap Q = \varnothing$, which by lemma 8.2 implies that $\tilde{Q} \cap \tilde{W} = \varnothing$. Thus we can rewrite expression (8:12) in the following way:

$$\left|\tilde{P}\right| - \left| \bigcup_{\substack{Q \in \boldsymbol{P}: p^{(1)} \notin Q,\, p^{(2)} \in Q \\ W \in \boldsymbol{P}: p^{(2)} \notin W,\, p^{(1)} \in W}} \tilde{P} \cap \tilde{Q} \cap \tilde{W} \right|$$

$$= \left|\tilde{P}\right| - \left| \bigcup_{\gamma \in \Gamma \setminus P} \tilde{P} \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\}) \right|. \tag{8:13}$$

We denote $\tilde{P} \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\})$ by $\mathrm{G}(\gamma)$, so that we can write (8:13) as:

$$= \left|\tilde{P}\right| - \left| \bigcup_{\gamma \in \Gamma \setminus P} \tilde{P} \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\}) \right|$$

$$= \left|\tilde{P}\right| - \left| \bigcup_{\gamma \in \Gamma \setminus P} \mathrm{G}(\gamma) \right| =$$

$$= \left|\tilde{P}\right| -$$

$$-\left(\sum_{\gamma \in \Gamma \backslash P} |G(\gamma)| - \sum_{\substack{\gamma^{(1)}, \gamma^{(2)} \in \Gamma \backslash P \\ \gamma^{(1)} \neq \gamma^{(2)}}} \left|G(\gamma^{(1)}) \cap G(\gamma^{(2)})\right| + \ldots \right). \qquad (8:14)$$

We will stop to examine $G(\gamma^{(1)}) \cap G(\gamma^{(2)})$ in expression (8:14) when $\gamma^{(1)} \neq \gamma^{(2)}$.

$$G(\gamma^{(1)}) \cap G(\gamma^{(2)})$$

$$= \tilde{P} \cap \mathrm{FS}(\{p^{(2)}, \gamma^{(1)}\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma^{(1)}\})$$

$$\cap \mathrm{FS}(\{p^{(2)}, \gamma^{(2)}\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma^{(2)}\})$$

$$\subseteq \mathrm{FS}(\{p^{(1)}, \gamma^{(1)}\}) \cap \mathrm{FS}(\{p^{(2)}, \gamma^{(2)}\}) = \varnothing$$

The last equality is due to lemma 8.2. This implies that only the first sum inside the parenthesis in expression (8:14) can be non-zero, so we can rewrite (8:14) as follows:

$$\left|\tilde{P}\right| - \left(\sum_{\gamma \in \Gamma \backslash P} |G(\gamma)| - \sum_{\substack{\gamma^{(1)}, \gamma^{(2)} \in \Gamma \backslash P \\ \gamma^{(1)} \neq \gamma^{(2)}}} \left|G(\gamma^{(1)}) \cap G(\gamma^{(2)})\right| + \ldots \right) =$$

$$= \left|\tilde{P}\right| - \sum_{\gamma \in \Gamma \setminus P} |\mathrm{G}(\gamma)|$$

$$= \left|\tilde{P}\right| - \sum_{\gamma \in \Gamma \setminus P} \left|\tilde{P} \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\})\right|$$

$$= \left|\tilde{P}\right|$$

$$- \sum_{\gamma \in \Gamma \setminus P} \left|\mathrm{FS}(\{p^{(1)}, p^{(2)}\}) \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\})\right| \ . \quad (8:15)$$

For a single term in the sum in (8:15) we have

$$\left|\mathrm{FS}(\{p^{(1)}, p^{(2)}\}) \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\})\right| . \quad (8:16)$$

Examine a single coordinate for the three codewords γ, $p^{(1)}$ and $p^{(2)}$. The following combinations are possible in this coordinate:

$$\begin{array}{ll} p^{(1)} & 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1 \\ p^{(2)} & 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ \gamma & 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1 \end{array} \ . \quad (8:17)$$

These different combinations yield the following result for the corresponding feasible set coordinates, where '*x*' means that the coordinate is detectable:

$$\begin{array}{ll} \mathrm{FS}(\{p^{(1)}, p^{(2)}\}) & 0\ 0\ x\ x\ x\ x\ 1\ 1 \\ \mathrm{FS}(\{p^{(1)}, \gamma\}) & 0\ x\ 0\ x\ x\ 1\ x\ 1 \\ \mathrm{FS}(\{p^{(2)}, \gamma\}) & 0\ x\ x\ 1\ 0\ x\ x\ 1 \end{array} \ . \quad (8:18)$$

That a coordinate is detectable for a set of codewords means that this set of codewords has the ability to create both '0' and '1' in this coordinate. Since we are looking for the intersection of what three sets of codewords can construct (see (8:16)), the coordinate under consideration must, to be in the intersection, be the following for the different combinations in (8:17):

$$0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ \cdot \tag{8:19}$$

That is, in every case the coordinate is specified by the intersection of the feasible sets, and in no case are there any contradictions between the different feasible sets in (8:18).

Since the coordinate was chosen arbitrarily, this holds for every coordinate, so there is exactly one word in the intersection in (8:16). Since also γ was chosen arbitrarily, this holds for every $\gamma \in \Gamma \backslash P$ and (8:15) evolves into

$$\left|\tilde{P}\right| - \sum_{\gamma \in \Gamma \backslash P} \left|\mathrm{FS}(\{p^{(1)}, p^{(2)}\}) \cap \mathrm{FS}(\{p^{(2)}, \gamma\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\})\right|$$

$$= \left|\tilde{P}\right| - \sum_{\gamma \in \Gamma \backslash P} 1 \quad = \left|\tilde{P}\right| - (|\Gamma| - |P|) \quad = \left|\tilde{P}\right| - |\Gamma| + 2\,,$$

so that we finally get that when $|P| = 2$ we have

$$N = \left|\tilde{P}\right| - |\Gamma| + 2\,. \tag{8:20}$$

We will now examine the numerator N in the sum in (8:9) when $|P| = 1$, letting $P = \{p\}$.

$$N = \left| \bigcup_{x \in P} (\tilde{P} \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \tilde{Q}) \right| = \left| \tilde{P} \setminus \bigcup_{Q \in \boldsymbol{P}: p \notin Q} \tilde{Q} \right|$$

$$= |\tilde{P}| - \left| \tilde{P} \cap \bigcup_{Q \in \boldsymbol{P}: p \notin Q} \tilde{Q} \right| = 1 - \left| \bigcup_{Q \in \boldsymbol{P}: p \notin Q} (\tilde{P} \cap \tilde{Q}) \right| \tag{8:21}$$

We have that $P, Q \in \boldsymbol{P}$, and $p \notin Q$ so $P \cap Q = \varnothing$. By lemma 8.2 this implies that $\tilde{P} \cap \tilde{Q} = \varnothing$, so we find that when $|P| = 1$ we have

$$N = 1 - \left| \bigcup_{Q \in \boldsymbol{P}: p \notin Q} (\tilde{P} \cap \tilde{Q}) \right| = 1 . \tag{8:22}$$

Using expression (8:20) and (8:22) in the performance measure yields:

$$\mu(\Gamma) = \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}} \frac{\left| \bigcup_{x \in P} (\mathrm{FS}(P) \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \mathrm{FS}(Q)) \right|}{|\mathrm{FS}(P)|}$$

$$= \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}: |P| = 1} \frac{\left| \bigcup_{x \in P} (\mathrm{FS}(P) \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \mathrm{FS}(Q)) \right|}{|\mathrm{FS}(P)|} +$$

$$+ \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}: |P| = 2} \frac{\left| \bigcup_{x \in P} (\mathrm{FS}(P) \setminus \bigcup_{Q \in \boldsymbol{P}: x \notin Q} \mathrm{FS}(Q)) \right|}{|\mathrm{FS}(P)|}$$

$$= \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}: |P| = 1} \frac{1}{|\mathrm{FS}(P)|} + \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}: |P| = 2} \frac{|\mathrm{FS}(P)| - |\Gamma| + 2}{|\mathrm{FS}(P)|} =$$

$$= \frac{1}{|\boldsymbol{P}|}\left(\left(\sum_{P \in \boldsymbol{P}:|P|=1} 1\right) + \left(\sum_{P \in \boldsymbol{P}:|P|=2} \left(1 - \frac{|\Gamma|-2}{|\mathrm{FS}(P)|}\right)\right)\right)$$

$$= \frac{1}{|\boldsymbol{P}|}(|\{P \in \boldsymbol{P}:|P| = 1\}|) +$$

$$+ \frac{1}{|\boldsymbol{P}|}\left(|\{P \in \boldsymbol{P}:|P| = 2\}| - (|\Gamma|-2) \sum_{P \in \boldsymbol{P}:|P|=2} \frac{1}{|\mathrm{FS}(P)|}\right)$$

$$= \frac{|\{P \in \boldsymbol{P}:|P| = 1\}| + |\{P \in \boldsymbol{P}:|P| = 2\}|}{|\boldsymbol{P}|} - \frac{|\Gamma|-2}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P|=2} \frac{1}{|\mathrm{FS}(P)|}$$

$$= 1 - \frac{|\Gamma|-2}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P|=2} \frac{1}{|\mathrm{FS}(P)|}. \qquad (8:23)$$

In this expression $|\Gamma| = M$, the size of the code, and $|\boldsymbol{P}| = \binom{M}{2} + \binom{M}{1}$, the number of different codeword subsets of size 1 or 2. The only things needed to actually calculate $\mu(\Gamma)$ are the sizes of the different feasible sets, $|\mathrm{FS}(P)|$. These are dependent on the code, but knowing the code we can calculate them as $|\mathrm{FS}(P)| = 2^{d_H(p^{(1)}, p^{(2)})}$, where $d_H(p^{(1)}, p^{(2)})$ is the Hamming distance between the two elements in *P*, or equivalently, the number of coordinates detectable to *P*.

8.4 Code Constructions

8.4.1 M/2-codes

Definition 8.3: *M/2-codes* are codes Γ with an even number of codewords, $|\Gamma| \geq 4$, and such that the code matrix contains all column types for which the weight is $M/2$ and the most significant bit is zero. Further, all column types are repeated the same number of times, δ.

The M/2-codes can be characterized by M, the number of codewords in the code, and δ, the number of times each column in the code matrix is repeated. We get that $n = \delta\binom{M-1}{M/2}$.

Lemma 8.3: If Γ is an M/2-code then Γ is a PCS code.

Proof: Let Γ be an M/2-code and choose any four different codewords $a, \alpha, b, \beta \in \Gamma$ and let $\{a, \alpha\} = A$ and $\{b, \beta\} = B$. We have that $|A| = |B| = 2$ and $A \cap B = \varnothing$. There are now two cases, depending on whether or not the zero word is among a, α, b, β.

1. Assume that the zero codeword is among a, α, b, β, and without loss of generality assume that a is that codeword. For Γ to be a PCS code there must exist a coordinate i such that $\alpha_i = 0$ and $b_i = \beta_i = 1$. But by definition the code matrix of Γ contains all column types for which the weight is $M/2$ and the most significant bit is zero, so there will exist a column (coordinate) i in the code matrix such that $a_i = \alpha_i = 0$ and $b_i = \beta_i = 1$.
2. Assume that the zero codeword is not among a, α, b, β. By definition the code matrix of Γ contains all column types for which the weight is $M/2$ and the most significant bit is zero, so there will exist a column (coordinate) i in the code matrix such that $a_i = \alpha_i \neq b_i = \beta_i$.

❑

Lemma 8.4: If Γ is an M/2-code and $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$, $\gamma^{(i)} \neq \gamma^{(j)}$, then the Hamming distance

$$d_H(\gamma^{(i)}, \gamma^{(j)}) = \delta\binom{M-2}{M/2-1}.$$

Proof: Choose $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$. We will consider two cases, depending on whether or not the zero codeword is one of $\gamma^{(i)}, \gamma^{(j)}$.

1. Assume, without loss of generality, that $\gamma^{(i)}$ is the zero codeword. The Hamming distance between the codewords is then equal to the number of ones in $\gamma^{(j)}$. In coordinates in which $\gamma^{(j)}$ is equal to one there are $M/2$ - 1 further ones to distribute among the other M - 2 rows (M - 2 since $\gamma^{(i)} = 0$). This can be done in $\binom{M-2}{M/2-1}$ ways. Since each column starting with a zero, and of weight $M/2$ exists and is repeated δ times in the code matrix, each of these $\binom{M-2}{M/2-1}$ ways corresponds to δ coordinates, so when $\gamma^{(i)}$ is the zero codeword we have that $d_H(\gamma^{(i)}, \gamma^{(j)}) = \delta\binom{M-2}{M/2-1}$.
2. Assume that the zero codeword is not among $\gamma^{(i)}, \gamma^{(j)}$. The Hamming distance between the codewords is then equal to the number of coordinates k in which $\gamma^{(i)}$ is not equal to $\gamma^{(j)}$. This can happen in two different ways: either $\gamma_k^{(i)} = 0$ and $\gamma_k^{(j)} = 1$, or the other way around. When $\gamma_k^{(i)} = 0$ and $\gamma_k^{(j)} = 1$ there are $M/2$ - 1 further ones to distribute among the other $M-3$ rows (we cannot place them in $\gamma^{(i)}$, $\gamma^{(j)}$ or the zero codeword). This can be done in $\binom{M-3}{M/2-1}$ ways, each of which corresponds to δ coordinates. In the same way we get that there are $\binom{M-3}{M/2-1}$ ways to place the ones when $\gamma_k^{(i)} = 1$ and $\gamma_k^{(j)} = 0$, each of which corresponds to δ coordinates. These cases taken together means that when neither of $\gamma^{(i)}, \gamma^{(j)}$ is the zero codeword, we have that $d_H(\gamma^{(i)}, \gamma^{(j)}) = 2\delta\binom{M-3}{M/2-1}$.

$$2\delta\binom{M-3}{M/2-1} = \delta\frac{M-2}{M/2-1} \times \frac{(M-3)!}{(M/2-1)!(M/2-2)!} =$$

$$= \delta \frac{(M-2)!}{(M/2-1)!(M/2-1)!} = \delta \binom{M-2}{M/2-1}$$

❑

Using lemma 8.4 we get that the size of the feasible set for any set containing two different codewords of a M/2-code is

$$|\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})| = 2^{\delta \binom{M-2}{M/2-1}}. \tag{8:24}$$

Using this result in expression (8:23) we find the performance of M/2-codes to be:

$$\mu(\Gamma) = 1 - (M-2) \frac{\binom{M}{2}}{\binom{M}{2} + \binom{M}{1}} 2^{-\delta \binom{M-2}{M/2-1}}$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} 2^{-\delta \binom{M-2}{M/2-1}}. \tag{8:25}$$

To see how this relates to the code length, n, we insert the relation $\delta = n / \binom{M-1}{M/2}$ in (8:25) and get:

$$\mu(\Gamma) = 1 - \frac{(M-2)(M-1)}{M+1} 2^{-n \binom{M-2}{M/2-1} / \binom{M-1}{M/2}} =$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} 2^{-\frac{M}{2(M-1)}n}. \qquad (8:26)$$

Computing values of $1-\mu(\Gamma)$ for some different values on M and δ yields the following table.

M	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$
4	3 0.3	6 0.075	9 1.87×10^{-2}	12 4.69×10^{-3}	15 1.17×10^{-3}	18 2.93×10^{-4}	21 7.32×10^{-5}
6	10 4.46×10^{-2}	20 6.98×10^{-4}	30 1.09×10^{-5}	40 1.70×10^{-7}	50 2.66×10^{-9}	60 4.16×10^{-11}	70 6.50×10^{-13}
8	35 4.45×10^{-6}	70 4.24×10^{-12}	105 4.05×10^{-18}	140 3.86×10^{-24}	175 3.68×10^{-30}	210 3.51×10^{-36}	245 3.35×10^{-42}
10	126 5.54×10^{-21}	252 4.70×10^{-42}	378 3.98×10^{-63}	504 3.37×10^{-84}	630 2.85×10^{-105}	756 2.42×10^{-126}	882 2.0×10^{-147}

Table 8.1: *n and* $1-\mu(\Gamma)$ *for some different values on M and* δ.

8.4.2 Column Weight Codes

Definition 8.4: A *column weight code* is a code Γ, $|\Gamma| \geq 4$, such that the code matrix is made up of every column type of weight w, $2 \leq w \leq \lfloor M/2 \rfloor$, each repeated the same number of times, δ ($\delta \geq 1$). For column weight codes we get the relation $n = \delta\binom{M}{w}$.

Lemma 8.5: If Γ is a column weight code then Γ is a PCS code.

Proof: Analogous to the proof of lemma 8.3.

The column weight codes can be characterized by *M*, the number of code-words in the code, *w*, the weight of the columns in the code matrix, and δ, the number of times each column in the code matrix is repeated.

Lemma 8.6: Let Γ be a column weight code and let $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$, $\gamma^{(i)} \neq \gamma^{(j)}$. Then

$$d_H(\gamma^{(i)}, \gamma^{(j)}) = 2\delta\binom{M-2}{w-1}. \tag{8:27}$$

Proof: Let Γ be a column weight code and let $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$, $\gamma^{(i)} \neq \gamma^{(j)}$. We will count in how many coordinates k it holds that $\gamma_k^{(i)} \neq \gamma_k^{(j)}$. There are two cases, $\gamma_k^{(i)} = 0$ and $\gamma_k^{(i)} = 1$.

We start by examining the case when $\gamma_k^{(i)} = 0$ and $\gamma_k^{(i)} = 1$. There are $w-1$ ones left to distribute between $M-2$ codewords. This can be done in $\binom{M-2}{w-1}$ different ways. Since the code matrix consists of *all* different column types of weight w, each of these $\binom{M-2}{w-1}$ different ways corresponds to a different column type, each of which is repeated δ times. In all this means that there are $\delta\binom{M-2}{w-1}$ coordinates in which $\gamma_k^{(i)} = 0$ and $\gamma_k^{(j)} = 1$.

In the same way we get that there are $\delta\binom{M-2}{w-1}$ coordinates in which $\gamma_k^{(i)} = 1$ and $\gamma_k^{(j)} = 0$.

The two cases taken together prove the lemma.

❑

Since column weight codes are PCS codes we can use expression (8:23) to get the performance, using that for any $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$, $\gamma^{(i)} \neq \gamma^{(j)}$ we have

$$\left|\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})\right| = 2^{2\delta\binom{M-2}{w-1}}.$$

In the same way that we found (8:25), we get

$$\mu(\Gamma) = 1 - \frac{(M-2)(M-1)}{M+1} 2^{-2\delta\binom{M-2}{w-1}}. \qquad (8:28)$$

To see how this relates to the code length, n, we insert the relation $\delta = n \diagup \binom{M}{w}$ in (8:28) and get:

$$\mu(\Gamma) = 1 - \frac{(M-2)(M-1)}{M+1} 2^{-2n\binom{M-2}{w-1}\diagup\binom{M}{w}}$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} 2^{-2\left(\frac{w(M-w)}{M(M-1)}\right)n}. \qquad (8:29)$$

Computing values for $1-\mu(\Gamma)$ and n for some different values on M, w and δ yields the following tables.

w	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$
2	6 0.075	12 4.69×10^{-3}	18 2.93×10^{-4}	24 1.83×10^{-5}	30 1.14×10^{-6}	36 7.15×10^{-8}	42 4.47×10^{-9}

Table 8.2: *n and* $1-\mu(\Gamma)$ *for some different values on* δ *when M = 4.*

w	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$
2	10 3.13×10^{-2}	20 4.89×10^{-4}	30 7.63×10^{-6}	40 1.19×10^{-7}	50 1.86×10^{-9}	60 4.37×10^{-11}	70 4.55×10^{-13}

Table 8.3: *n and* $1-\mu(\Gamma)$ *for some different values on* δ *when M = 5.*

w	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$
2	15 1.12×10^{-2}	30 4.36×10^{-5}	45 1.70×10^{-7}	60 6.65×10^{-10}	75 2.60×10^{-12}	90 1.06×10^{-14}	105 3.97×10^{-17}
3	20 6.98×10^{-4}	40 1.70×10^{-7}	60 4.16×10^{-11}	80 1.02×10^{-14}	100 2.48×10^{-18}	120 6.05×10^{-22}	140 2.07×10^{-25}

Table 8.4: *n and* $1-\mu(\Gamma)$ *for some different values on* δ *when M = 6.*

We can see that M/2-codes are similar to column weight codes when *M* is even and *w* = *M*/2. The construction of M/2-codes is such that if the δ for the M/2-code is chosen as twice that of the δ of the column weight code, *M* even and *w* = *M*/2, then the M/2-code and the column weight code are equivalent.

8.4.3 Identity Matrix Codes

Definition 8.5: A code Γ is an *identity matrix code* if its code matrix is made up of every column of weight one, each repeated δ times ($\delta \geq 1$).

Identity matrix codes are not PCS codes, so we will find their performance in another way. We start with the case $|P| = 2$ and let Γ be an identity matrix code of size *M*. By definition 8.5 we get that $n = \delta M$. We will examine $\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})$ for $\gamma^{(i)}, \gamma^{(j)} \in \Gamma$ when $\gamma^{(i)} \neq \gamma^{(j)}$. By the construction we have that

$$\begin{aligned}\gamma^{(i)} &= (0\ldots01\ldots10\ldots00\ldots00\ldots0)\\ \gamma^{(j)} &= (0\ldots00\ldots00\ldots01\ldots10\ldots0)\end{aligned},$$

and thus that $\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\}) = (0\ldots0x\ldots x0\ldots0x\ldots x0\ldots0)$, where *x* means that this position can be either ‘0’ or ‘1’. If a ‘1’ is chosen in any of these places it is possible to trace one of $\gamma^{(i)}$ and $\gamma^{(j)}$, since each codeword is unique in the placement of ones. The size of $\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})$ is

$|\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})| = 2^{2\delta}$, out of which exactly one word, the zero-word, is not i.p.p.

Thus $|\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\}) \cap \mathrm{IPP}(\Gamma)| / |\mathrm{FS}(\{\gamma^{(i)}, \gamma^{(j)}\})| = (2^{2\delta} - 1)/2^{2\delta} = 1 - 2^{-2\delta}$. This holds for any choice of $\{\gamma^{(i)}, \gamma^{(j)}\} \in \boldsymbol{P}$, $\gamma^{(i)} \neq \gamma^{(j)}$. In the case that the collusion consists of only one pirate the feasible set consists of only this pirate's codeword, something no collusion not containing this pirate can create. Thus, in this case no word in the feasible set is untraceable, so $|\mathrm{FS}(\{\gamma^{(i)}\}) \cap \mathrm{IPP}(\Gamma)| / |\mathrm{FS}(\{\gamma^{(i)}\})| = 1$. Taken together this means that:

$$\mu(\Gamma) = \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|}$$

$$= \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 1} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|} + \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|}$$

$$= \frac{1}{|\boldsymbol{P}|}(|\{P \in \boldsymbol{P}:|P| = 1\}| + |\{P \in \boldsymbol{P}:|P| = 2\}|(1 - 2^{-2\delta}))$$

$$= \frac{|\{P \in \boldsymbol{P}:|P| = 1\}| + |\{P \in \boldsymbol{P}:|P| = 2\}| - |\{P \in \boldsymbol{P}:|P| = 2\}|2^{-2\delta}}{|\boldsymbol{P}|}$$

$$= \frac{|\boldsymbol{P}| - |\{P \in \boldsymbol{P}:|P| = 2\}|2^{-2\delta}}{|\boldsymbol{P}|} = 1 - \frac{\binom{M}{2}}{\binom{M}{1} + \binom{M}{2}} 2^{-2\delta}$$

$$= 1 - \frac{M - 1}{M + 1} 2^{-2\delta}. \qquad (8{:}30)$$

We see that (8:30) decreases towards $1 - 2^{-2\delta}$ as M increases.

To see how (8:30) relates to the code length n, we insert the relation $\delta = n/M$ and get:

$$\mu(\Gamma) = 1 - \frac{M-1}{M+1} 2^{-2n/M}. \tag{8:31}$$

For some different values on M and δ we get the following table:

M	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$	$\delta = 6$	$\delta = 7$
3	3 0.125	6 3.13×10^{-2}	9 7.81×10^{-3}	12 1.95×10^{-3}	15 4.88×10^{-4}	18 1.22×10^{-4}	21 3.05×10^{-5}
4	4 0.15	8 3.75×10^{-2}	12 9.38×10^{-3}	16 2.34×10^{-3}	20 5.86×10^{-4}	24 1.46×10^{-4}	28 3.66×10^{-5}
5	5 0.167	10 4.17×10^{-2}	15 1.04×10^{-2}	20 2.60×10^{-3}	25 6.51×10^{-4}	30 1.63×10^{-4}	35 4.07×10^{-5}
6	6 0.179	12 4.46×10^{-2}	18 1.12×10^{-4}	24 2.79×10^{-3}	30 6.98×10^{-4}	36 1.74×10^{-4}	42 4.36×10^{-5}

Table 8.5: *n and* $1-\mu(\Gamma)$ *for some different values on M and* δ.

8.5 Optimal and Best Found Codes

Using the algebra systems Magma V2.7-2 and SYMMETRICA[1] we have done computer searches to find optimal codes for small values of n and M. This has been done by first finding a representative for every code equivalence class for a certain code size (n and M), and then evaluating each one using expression (8:5), finding the best ones. For larger sizes of codes we

1. The author wishes to express his gratitude towards Dr. Harald Fripertinger who kindly placed his code constructing SYMMETRICA program at the author's disposal.

have not been able to do complete searches over all codes. In these cases we have tried derivations of the constructions in section 8.4, random computer searches and manual constructions.

Table 8.6 below contains the performance of the best found codes so far of each size and length. The codes that have a recognizable structure have been marked with a tag that describes the structure (see below). Appendix B tabulates codes that yield these performances.

Shaded cells mean that for these sizes a complete search over all codes has been done and thus that the values of the performance measure are optimal for these code lengths and code sizes. The following tags are used to describe the code structure:

M2 M/2-code
CW Column weight code
IM Identity matrix code
Mx M/2-code with repeated or removed columns
Ix Identity matrix code with repeated or removed columns

M	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 7$	$n = 8$	$n = 9$	$n = 10$
3	0.1250^{IM}	0.0833^{Ix}	0.0521^{Ix}	0.0313^{IM}	0.0208^{Ix}	0.0130^{Ix}	0.0078^{IM}	0.0052^{Ix}
4	0.3000^{Ix}	0.1500^{IM}	0.1125^{Ix}	0.0750^{M2}	0.0500^{Mx}	0.0313^{Mx}	0.0188^{M2}	0.0125^{Mx}
5	0.8000	0.3000^{Ix}	0.1666^{IM}	0.1333^{Ix}	0.0969	0.0625	0.0453	0.0312^{CW}
6		0.6786	0.2857^{Ix}	0.1786^{IM}	0.1488^{Ix}	0.1071	0.0714^{Mx}	0.0446^{M2}
7		0.8393	0.5982	0.2768^{Ix}	0.1875^{IM}	0.1607^{Ix}	0.1362^{Ix}	0.1138^{Ix}
8		0.8750	0.7431	0.5417	0.2708^{Ix}	0.1944^{IM}	0.1701^{Ix}	0.1476^{Ix}
9		0.8889	0.7833	0.6525	0.5000	0.2667^{Ix}	0.2000^{IM}	0.1778^{Ix}
10		1	0.8045	0.6833	0.6273	0.4682	0.2636^{Ix}	0.2045^{IM}

Table 8.6: $1 - \mu(\Gamma)$ *for the best found codes of different sizes and lengths.*

8.6 Bounds

8.6.1 Lower Bound

We will use a randomly chosen code to construct a lower bound on the performance measure. Here, unlike in the random code model used in chapters 3 to 5, the random code is merely a tool to calculate the average performance of such a code.

Assume that we create the codewords in the fingerprinting code Γ randomly, with each bit chosen independently and with equal probability of '0' and '1'. Since we choose the code like this the performance will also be random. What we will do is to find a bound on the expected performance for this code. We will assume that $|\Gamma| \geq 2$.

With $\mu_{\text{PCS}}(\Gamma)$ we mean the formula for the performance of PCS codes in expression (8:23), applied to any code Γ, regardless of whether it is a PCS code or not. We must only keep in mind that if Γ is not a PCS code, then $\mu_{\text{PCS}}(\Gamma)$ is not necessarily equal to the code performance $\mu(\Gamma)$.

$$\mathrm{E}[\mu(\Gamma)] = \Pr(\Gamma \in \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\big|\Gamma \in \text{PCSC}] +$$

$$+ \Pr(\Gamma \notin \text{PCSC})\mathrm{E}[\mu(\Gamma)|\Gamma \notin \text{PCSC}]$$

$$\geq \Pr(\Gamma \in \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\big|\Gamma \in \text{PCSC}]$$

To continue we note that $\mu_{\text{PCS}}(\Gamma) \leq 1$ for all Γ, so we get the following:

$$\mathrm{E}[\mu(\Gamma)] \geq \Pr(\Gamma \in \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\big|\Gamma \in \text{PCSC}] \geq$$

$$\geq \Pr(\Gamma \in \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\,|\,\Gamma \in \text{PCSC}] +$$

$$+ \Pr(\Gamma \notin \text{PCSC})(\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\,|\,\Gamma \notin \text{PCSC}] - 1)$$

$$= \Pr(\Gamma \in \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\,|\,\Gamma \in \text{PCSC}] +$$

$$+ \Pr(\Gamma \notin \text{PCSC})\mathrm{E}[\mu_{\text{PCS}}(\Gamma)\,|\,\Gamma \notin \text{PCSC}] - \Pr(\Gamma \notin \text{PCSC})$$

$$= \mathrm{E}[\mu_{\text{PCS}}(\Gamma)] - \Pr(\Gamma \notin \text{PCSC})\,. \tag{8:32}$$

To continue we need to find expressions for the probability of a randomly chosen code Γ not being a PCS code, and for the expected value of $\mu_{\text{PCS}}(\Gamma)$.

For Γ not to be a PCS code, there must be at least two pairs of codewords $A = \{a, \alpha\}$, $B = \{b, \beta\}$ in Γ, $A \cap B = \varnothing$, such that there are no coordinates i in which $a_i = \alpha_i \neq b_i = \beta_i$ (see definition 8.2). The probability of $a_i = \alpha_i \neq b_i = \beta_i$ *not* holding in a certain coordinate i, for two certain pairs of codewords A and B, is 7/8. (There are 16 combinations of bit values, and only $1 = 1 \neq 0 = 0$ and $0 = 0 \neq 1 = 1$ makes $a_i = \alpha_i \neq b_i = \beta_i$ hold.) The probability of $a_i = \alpha_i \neq b_i = \beta_i$ not holding in any of the n coordinates is $(7/8)^n$. The probability of this happening at least once in Γ is upper bounded by $(7/8)^n \times$ (number of ways of choosing A and B). A can be chosen in $\binom{M}{2}$ ways, and B in $\binom{M-2}{2}$ ways. But the order in which we choose A and B does not matter, so the number of different ways of choosing A and B is $(1/2)\binom{M}{2}\binom{M-2}{2} = \frac{M!}{8(M-4)!}$. Combining these results we get:

$$\Pr(\Gamma \notin \text{PCSC}) \leq \min\left(\frac{M!(7/8)^n}{8(M-4)!}, 1\right), \tag{8:33}$$

where the minimum-operation comes from the fact that a probability is always less than or equal to one.

We will now examine $\mathrm{E}[\mu_{\mathrm{PCS}}(\Gamma)]$. From expression (8:23) we get:

$$\mathrm{E}[\mu_{\mathrm{PCS}}(\Gamma)] = \mathrm{E}\left[1 - \frac{|\Gamma| - 2}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{1}{|\mathrm{FS}(P)|}\right]$$

$$= 1 - \frac{(M-2)}{\binom{M}{2} + \binom{M}{1}} \binom{M}{2} \mathrm{E}\left[\frac{1}{|\mathrm{FS}(P)|} \middle| |P| = 2\right]$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} \mathrm{E}\left[\frac{1}{|\mathrm{FS}(P)|} \middle| |P| = 2\right]. \tag{8:34}$$

Letting $P = \{x, y\}$, and letting $d_H(x, y)$ denote the Hamming distance between *x* and *y*, we can express (8:34) in the following way:

$$1 - \frac{(M-2)(M-1)}{M+1} \mathrm{E}\left[\frac{1}{|\mathrm{FS}(P)|} \middle| |P| = 2\right]$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} \mathrm{E}\left[\frac{1}{2^{d_H(x,y)}}\right]$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} \sum_{d=0}^{n} \Pr(d_H(x, y) = d) 2^{-d} =$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} \sum_{d=0}^{n} \binom{n}{d} 2^{-d} 2^{-(n-d)} 2^{-d}$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} \sum_{d=0}^{n} \binom{n}{d} 4^{-d} 2^{-(n-d)}$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} (4^{-1} + 2^{-1})^n$$

$$= 1 - \frac{(M-2)(M-1)}{M+1} (3/4)^n . \qquad (8:35)$$

Using expressions (8:33) and (8:35) in (8:32) we get:

$$\mathrm{E}[\mu(\Gamma)] \geq \mathrm{E}[\mu_{\mathrm{PCS}}(\Gamma)] - \Pr(\Gamma \notin \mathrm{PCSC})$$

$$\geq 1 - \frac{(M-2)(M-1)}{M+1} (3/4)^n - \min\left(\frac{M!(7/8)^n}{8(M-4)!}, 1 \right). \qquad (8:36)$$

Since we have a lower bound on the expected performance of a random code, with the expectation taken over all possible codes, there must exist codes that have a performance greater than this lower bound. Thus the bound on the expected performance of random codes defines a non-trivial lower bound on the performance of the optimal code of any given length and size (except in the cases when expression (8:36) is non-positive).

We convert (8:36) to an upper bound on $\mathrm{E}[1-\mu(\Gamma)]$.

$$\mathrm{E}[1-\mu(\Gamma)] = 1-\mathrm{E}[\mu(\Gamma)]$$

$$\leq \frac{(M-2)(M-1)}{M+1}(3/4)^n + \min\left(\frac{M!(7/8)^n}{8(M-4)!}, 1\right)$$

When n is large enough we can write this as:

$$\mathrm{E}[1-\mu(\Gamma)] \leq \frac{(M-2)(M-1)}{M+1}(3/4)^n + \frac{M!(7/8)^n}{8(M-4)!}. \tag{8:37}$$

(8:37) decreases $\mathrm{O}((7/8)^n)$ in n when n grows. For comparison, $1-\mu(\Gamma)$ for M/2-codes decreases $\mathrm{O}((1/2^{(M/(2(M-1)))})^n)$, which is faster than $\mathrm{O}((1/\sqrt{2})^n)$, regardless of M. Thus, $1-\mu(\Gamma)$ for M/2-codes decreases faster than the expected value of $1-\mu(\Gamma)$ for random codes.

For identity matrix codes we have that $1-\mu(\Gamma)$ decreases $\mathrm{O}((1/2^{(2/M)})^n)$. This is faster than the expected value of $1-\mu(\Gamma)$ for random codes only if $M \leq 10$, otherwise it is slower. Obviously identity matrix codes are worse than the average, random code when the number of codewords is greater than 10.

8.6.2 Upper Bound

We restate the definition of the performance measure in the case $c = 2$:

$$\mu(\Gamma) = \frac{1}{|\boldsymbol{P}|}\sum_{P \in \boldsymbol{P}} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|}. \tag{8:38}$$

In each FS(P) with $|P| = 2$ there is at least one word that is not i.p.p. To see this, let $P = \{x, y\} \in \boldsymbol{P}$, and let $z \in \Gamma$, $z \notin P$. Construct the word a by for each coordinate i choosing a_i to be zero if there are more zeros that ones in $\langle x_i, y_i, z_i \rangle$, and otherwise choosing a_i to be one. With this construction we get that $a \in \mathrm{FS}(\{x, y\})$, $a \in \mathrm{FS}(\{x, z\})$ and $a \in \mathrm{FS}(\{y, z\})$. Then, by definition 8.1, we see that $a \notin \mathrm{IPP}(\Gamma)$. Thus $|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)| \leq |\mathrm{FS}(P)| - 1$.

When $|P| = 1$ we can use the bound $|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)| \leq |\mathrm{FS}(P)|$.

Using this we get:

$$\mu(\Gamma) = \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|}$$

$$= \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 1} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|} + \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{|\mathrm{FS}(P) \cap \mathrm{IPP}(\Gamma)|}{|\mathrm{FS}(P)|}$$

$$\leq \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 1} 1 + \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{|\mathrm{FS}(P)| - 1}{|\mathrm{FS}(P)|}$$

$$= \frac{|\{P \in \boldsymbol{P}:|P| = 1\}|}{|\boldsymbol{P}|} + \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \left(1 - \frac{1}{|\mathrm{FS}(P)|}\right)$$

$$= \frac{|\{P \in \boldsymbol{P}:|P| = 1\}| + |\{P \in \boldsymbol{P}:|P| = 2\}|}{|\boldsymbol{P}|} - \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{1}{|\mathrm{FS}(P)|}$$

$$. = \frac{|\boldsymbol{P}|}{|\boldsymbol{P}|} - \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{1}{|\mathrm{FS}(P)|} = 1 - \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P| = 2} \frac{1}{|\mathrm{FS}(P)|}. \quad (8:39)$$

$|\mathrm{FS}(P)|$ is trivially upper bounded by the size of the whole space.

$$\mu(\Gamma) \leq 1 - \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P|=2} \frac{1}{|\mathrm{FS}(P)|} \leq 1 - \frac{1}{|\boldsymbol{P}|} \sum_{P \in \boldsymbol{P}:|P|=2} 2^{-n}$$

$$= 1 - \frac{|\{P \in \boldsymbol{P}:|P|=2\}|}{|\boldsymbol{P}|} 2^{-n} = 1 - \frac{\binom{M}{2}}{\binom{M}{1}+\binom{M}{2}} 2^{-n}$$

$$= 1 - \frac{M-1}{M+1} 2^{-n} \tag{8:40}$$

We convert this bound to a lower bound on $1 - \mu(\Gamma)$ and get:

$$1 - \mu(\Gamma) \geq \frac{M-1}{M+1} 2^{-n}. \tag{8:41}$$

It is interesting to examine expression (8:39). In the special case that we are studying, with exactly two pirates, we have that $|\mathrm{FS}(P)| = 2^{d_H(p^{(1)}, p^{(2)})}$, when $P = \{p^{(1)}, p^{(2)}\}$, $p^{(1)} \neq p^{(2)}$. We thus have a strong connection to traditional coding theory, where the Hamming distance is the measure considered. Since the sum is taken over terms $1/|\mathrm{FS}(P)|$, the codeword pairs with the smallest Hamming distances will have a strong impact on the size of the sum. Judging from this we would like to have codes with a large minimum distance, which is probably the most studied code property in traditional coding theory.

8.7 Comparing the Codes

In figures 8.7 and 8.8 below (one minus) the performance of the different code classes is plotted The values are taken from tables 8.1, 8.4, 8.5 and 8.6 and presented in graphical form for ease of comparison. In the figures are also included the bounds from section 8.6.

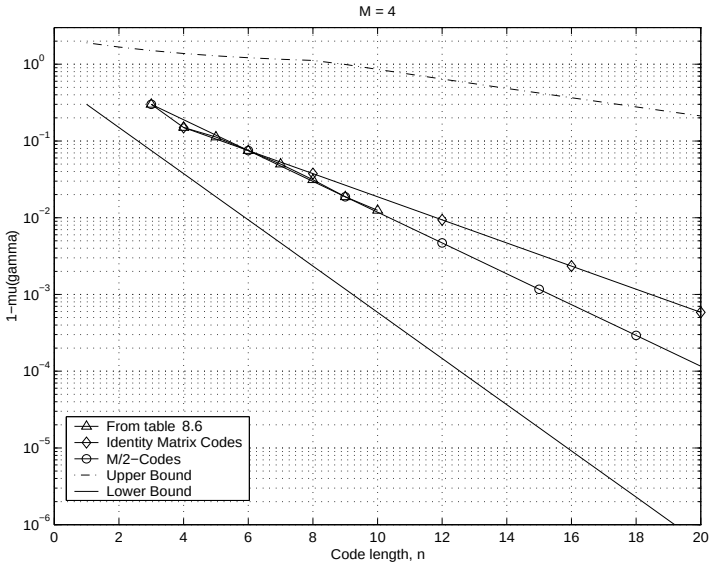

Figure 8.7: *The performance of the different code classes for M = 4.*

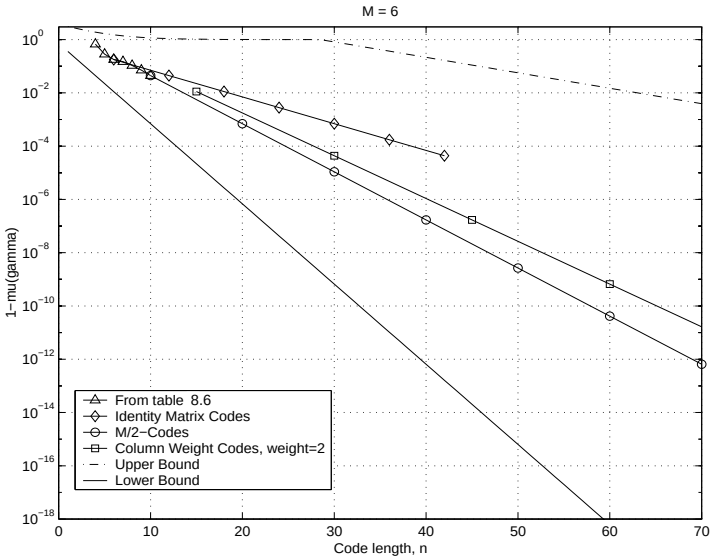

Figure 8.8: *The performance of the different code classes for M = 6.*

8.8 Tracing Probability and Performance

From a practical point of view we are interested in some kind of error probability of fingerprinting systems. When using codes of the types discussed in this chapter it is possible for the distributor to tell if a pirate can be correctly accused (the illegal word is i.p.p.) or if there is an uncertainty as to who is guilty (the illegal word is not i.p.p.). We will therefore assume that in the latter case the distributor will choose not to accuse anybody and instead accept that the tracing failed. We then get tracing probability instead of error probability as the "natural" performance measure. Should the distributor for some reason want to accuse somebody for *every* illegal word that is found, by guessing when it is not clear who is guilty, the error probability would be less than, or equal to the tracing failure probability (an error is made only if the word is not i.p.p. *and* the guess is wrong).

Unfortunately there is no straight forward relation between tracing probability and the performance measure that we have introduced. However, in some

special cases we have been able to compare them and see how they differ. Below we will do so for two of the code classes introduced in this chapter. We will assume that the number of pirates $c = 2$.

8.8.1 Identity Matrix Codes

In much the same way as in the example in section 7.5 we will find the probability of the pirates creating a word that is not in IPP(Γ).

Any collusion of two pirates will detect 2δ positions. The only word in their feasible set that is not traceable to any of them is the all-zero word, so the best strategy they can use is to choose δ positions from each of their codewords (otherwise it is not possible for them to create the zero word at all). This can be done in $\binom{2\delta}{\delta}$ different ways, out of which only one yields the non-traceable all-zero word. Thus the probability of the collusion creating a word that is not i.p.p. is $1/\binom{2\delta}{\delta}$.

Using Stirling's formula (in this case taken from [MWS77]),

$$\sqrt{2\pi}k^{k+1/2}e^{-k+1/(12k)-1/(360k^3)} < k! < \sqrt{2\pi}k^{k+1/2}e^{-k+1/(12k)}$$

we will construct upper and lower bounds for $\binom{2\delta}{\delta}$.

$$\binom{2\delta}{\delta} = \frac{(2\delta)!}{\delta!\delta!}$$

$$< \frac{\sqrt{2\pi}(2\delta)^{2\delta+1/2}e^{-2\delta+1/(12(2\delta))}}{\sqrt{2\pi}\delta^{\delta+1/2}e^{-\delta+1/(12\delta)-1/(360\delta^3)}\sqrt{2\pi}\delta^{\delta+1/2}e^{-\delta+1/(12\delta)-1/(360\delta^3)}}$$

$$= \frac{2^{2\delta}\delta^{2\delta}\sqrt{2}\sqrt{\delta}e^{-2\delta}e^{1/(24\delta)}}{\sqrt{2\pi}\delta^{\delta}\sqrt{\delta}e^{-\delta}e^{1/(12\delta)}e^{-1/(360\delta^3)}\delta^{\delta}\sqrt{\delta}e^{-\delta}e^{1/(12\delta)}e^{-1/(360\delta^3)}} =$$

$$= \frac{2^{2\delta}}{\sqrt{\delta\pi}e^{1/(8\delta)}e^{-1/(180\delta^3)}}$$

$$\binom{2\delta}{\delta} = \frac{(2\delta)!}{\delta!\delta!}$$

$$> \frac{\sqrt{2\pi}(2\delta)^{(2\delta)+1/2}e^{-(2\delta)+1/(12(2\delta))-1/(360(2\delta)^3)}}{\sqrt{2\pi}\delta^{\delta+1/2}e^{-\delta+1/(12\delta)}\sqrt{2\pi}\delta^{\delta+1/2}e^{-\delta+1/(12\delta)}}$$

$$= \frac{2^{2\delta}\delta^{2\delta}\sqrt{2}\sqrt{\delta}e^{-2\delta}e^{1/(24\delta)}e^{-1/(2880\delta^3)}}{\sqrt{2\pi}\delta^{\delta}\sqrt{\delta}e^{-\delta}e^{1/(12\delta)}\delta^{\delta}\sqrt{\delta}e^{-\delta}e^{1/(12\delta)}}$$

$$= \frac{2^{2\delta}e^{-1/(2880\delta^3)}}{\sqrt{\delta\pi}e^{1/(8\delta)}}$$

Thus we have shown the bounds:

$$\frac{2^{2\delta}e^{-1/(2880\delta^3)}}{\sqrt{\delta\pi}e^{1/(8\delta)}} < \binom{2\delta}{\delta} < \frac{2^{2\delta}}{\sqrt{\delta\pi}e^{1/(8\delta)}e^{-1/(180\delta^3)}}. \tag{8:42}$$

This means that $1/\binom{2\delta}{\delta}$, and thus the probability of the pirates creating a word that is not i.p.p. decreases approximately as $\sqrt{\delta\pi}2^{-2\delta}$ with growing δ.

From (8:30) we have that for identity matrix codes $1-\mu(\Gamma) = \frac{M-1}{M+1}2^{-2\delta}$, that is, $1-\mu(\Gamma)$ decreases approximately as $2^{-2\delta}$ with growing δ. We can see that what differs between the two approximated expressions is the factor $\sqrt{\delta\pi}$. This factor grows slowly compared to the rate of decrease of $2^{-2\delta}$, so

the behaviour of the tracing probability for the strongest pirate strategy and the value of the performance is similar for the identity matrix codes.

8.8.2 M/2-codes

Let Γ be an M/2-code where each different column type is repeated δ times. Due to the symmetry of the code we can, without loss of generality, assume that the collusion P consists of the all-zero codeword, $p^{(1)}$, and one more codeword, $p^{(2)}$. We further assume that the coordinates are ordered so that the zeroes in $p^{(2)}$ are to the left and the ones to the right, so that $p^{(2)} = (0\ldots01\ldots1)$.

We will begin by finding out which elements in the feasible set of P are not i.p.p. Words in $\mathrm{FS}(P)$ that are not i.p.p. are words x such that $\bigcap_{Q:x\in\mathrm{FS}(Q)} Q = \varnothing$ (see expression (8:1)). For a given $x \in \mathrm{FS}(P)$ we examine the set of $Q \in \boldsymbol{P}$ such that $x \in \mathrm{FS}(Q)$. Then for every such x we have $x \in \mathrm{FS}(Q) \cap \mathrm{FS}(P)$, so lemma 8.2 (backwards) implies that $P \cap Q \neq \varnothing$. This means that every such Q is of the type $Q = \{p^{(1)}, \gamma\}$ or $Q = \{p^{(2)}, \gamma\}$. Thus, $x \in \mathrm{FS}(P)$ is not i.p.p. if and only if for any $\gamma \in \Gamma\backslash P$, we have

$$x \in \mathrm{FS}(\{p^{(1)}, p^{(2)}\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\}) \cap \mathrm{FS}(\{\gamma, p^{(2)}\}). \tag{8:43}$$

By reordering the coordinates we can get a code matrix for the codewords $p^{(1)}$, $p^{(2)}$ and γ (with the rows in that order) like that below.

$$\begin{array}{l} 0\ldots0\ 0\ldots0\ 0\ldots0\ 0\ldots0 \\ 0\ldots0\ 0\ldots0\ 1\ldots1\ 1\ldots1 \\ 0\ldots0\ 1\ldots1\ 0\ldots0\ 1\ldots1 \end{array} \tag{8:44}$$

Only the single word $x = (0\ldots00\ldots00\ldots01\ldots1)$ is in the intersection $\mathrm{FS}(\{p^{(1)}, p^{(2)}\}) \cap \mathrm{FS}(\{p^{(1)}, \gamma\}) \cap \mathrm{FS}(\{p^{(2)}, \gamma\})$. x can be characterized as having ones exactly in the coordinates in which both $p^{(2)}$ and γ contain

ones. This coordinate set has a one-to-one relation with γ. This can be seen by noting that the code Γ', formed by considering $\Gamma \backslash P$ only in the coordinates in which $p^{(2)}$ contains ones, is a code containing every different column of size $M-2$ with weight $M/2-1$, each repeated δ times, so each such codeword is unique and the coordinates of its ones is the coordinate set. Thus for each $\gamma \notin P$ there is exactly one word in $\mathrm{FS}(P)$ that is not i.p.p., so in all there are $M-2$ such words. Each such word has as a weight equal to the weight of the codewords in Γ' (which is a constant-weight code), and the ones are placed only among the coordinates in which $p^{(2)}$ contains ones. The weight of the codewords in Γ' is $(w_H(p^{(2)})(M/2-1))/(M-2) = w_H(p^{(2)})/2$. Thus, to create a word that is not i.p.p. the collusion P will have to choose $w_H(p^{(2)})/2$ coordinates from $p^{(1)}$ and $w_H(p^{(2)})/2$ from $p^{(2)}$. This can be done in

$$\binom{w_H(p^{(2)})}{w_H(p^{(2)})/2}$$

ways, out of which $M-2$ results in a word that is not i.p.p. The probability of the pirates creating a word that is not i.p.p. is then

$$\frac{M-2}{\binom{w_H(p^{(2)})}{w_H(p^{(2)})/2}} = \frac{M-2}{\left(\begin{matrix}\delta\binom{M-2}{M/2-1} \\ \frac{\delta}{2}\binom{M-2}{M/2-1}\end{matrix}\right)}.$$

Recalling (8:42) we will approximate $\binom{2\alpha}{\alpha}$ with $\frac{2^{2\alpha}}{\sqrt{\pi\alpha}}$ in the expression above.

$$\frac{M-2}{\binom{w_H(p^{(2)})}{w_H(p^{(2)})/2}} \approx \frac{(M-2)\sqrt{\pi\frac{\delta}{2}\binom{M-2}{M/2-1}}}{2^{\delta\binom{M-2}{M/2-1}}} =$$

$$= \sqrt{\pi\frac{\delta}{2}\binom{M-2}{M/2-1}}(M-2)2^{-\delta\binom{M-2}{M/2-1}} \qquad (8:45)$$

From (8:25) we have that for M/2-codes

$$1-\mu(\Gamma) = \frac{(M-1)}{M+1}(M-2)2^{-\delta\binom{M-2}{M/2-1}}.$$

The exponential factors dominates in both of these expressions, so the tracing probability for the strongest pirate strategy and the value of the performance is similar also for the M/2-codes.

8.9 Summary and Discussion

Sections 8.1 and 8.2 of this chapter are quite general. From the start of section 8.3 we assume that $c = 2$ and in lemma 8.2 we use the fact that we consider PCS codes.

The chapter thus contains two parts, both of which we find are of interest. The first is the performance measure $\mu(\Gamma)$, the definition of which is generally valid. The second part is the results for the special case $c = 2$. These consist of the simplified expression for $\mu(\Gamma)$ for PCS codes (expression (8:23)), the bounds, the code constructions and the codes found by computer search.

An interesting fact to note in table 8.6 is that the M/2-codes of size 4 and length 6 and 9 are both optimal. Unfortunately we have not been able to completely search the codes of length 10, size 6, so we do not know if the M/2-code is optimal also in this case.

The identity matrix codes are known to be optimal for $n = M$ up to size 7, and for greater sizes they are the best known. The discussion in the end of section 8.6 indicates that even the average random code is better than identity matrix

codes when $M \geq 11$ and n is large. An interesting question is how good identity matrix codes are in general when $n = M$.

We have seen that in the cases of the identity matrix codes and M/2-codes the performance and the tracing probability for the strongest pirate strategy behaves similarly. This is interesting, since if we could show that the performance and the tracing probability were closely related more generally, we could construct codes using the performance measure, knowing that they would yield a high tracing probability. Using the measure for code construction seems to be a practical application for it, since the performance measure is so much easier to handle than either error or tracing probability.

A special case is that all collusions are all equally probable, regardless of size, and that the pirates choose the illegal word randomly and uniformly among the words in their feasible set. Under these conditions the performance is equal to the probability that the collusion creates a word that is i.p.p., which thus deterministically can be traced to one of the pirates. Hence, in this special case there is a direct connection between the tracing probability and the performance.

What is nice with the kind of codes that we have examined in this chapter is that the distributor will know when (s)he finds the illegal word whether or not it can be traced correctly (assuming that the number of pirates is at most c). The distributor can choose not to accuse anybody if the illegal word is not i.p.p. and the result will be that no innocent user will ever be accused – there will be no probability of error.

Chapter 9

Future Work

The work presented in this thesis can be regarded as an exploration of the field of digital fingerprinting. We have seen that there exist interesting problems that are possible to state, analyse and reach conclusions about. Of course the exploration of this field is not finished with this thesis. There are still lots of open problems, and below we will describe some of them that we think are interesting.

In chapters 4 and 5 we avoided the problem of error probabilities for groups consisting in part of pirates and in part of innocent users. To make a complete analysis of any fingerprinting scheme aimed at dealing with collusions of pirates, whether for finding or testing, it is essential that also such groups are covered in terms of error probabilities.

Chapters 4 and 5 deal with the special case when the fingerprinting code is binary and random with each bit drawn independently with equal probability of ones and zeros. A natural generalization of this kind of random code would be to allow codes where the probabilities of ones and zeros were not equal. We could use a parameter to denote the probability of ones, for example $\alpha = \Pr(\gamma_i = 1)$, and examine how the different results depend on α.

The communication model of fingerprinting, with the fingerprints viewed as codewords to be sent, and the illegal fingerprint found viewed as the received

word, would be interesting to explore further. Is it possible to make an information-theoretical model of the fingerprinting problem, at least in special cases? Is there a fingerprinting equivalent to channel capacity? Basically the question is if it is possible to make a general, theoretical framework for what can and cannot be done with fingerprinting.

A step in this way would be if we could find better bounds on the performance of fingerprinting systems, both bounds for specific systems and general bounds. It would be of great help to be able to say that for a certain number of pirates, a certain number of users and a certain fingerprint length there exist systems that are at least *this* good, but none that are better than *this*.

Chapters 6 and 7 show that it is in principle easy to find good, or even optimal codes and strategies, both for the pirates and the distributor. Also in principle, it is easy to compute the tracing probability for any given code when both parties act optimally. Something very interesting would be to actually try to do this, at least for small problems.

Chapter 8 opens the door to a very rich field of problems about performance bounds and code constructions. There are connections between this performance measure and traditional coding theory that would be very interesting to explore. It would also be of great value to further examine how the performance measure is connected to the tracing probability. Can something be said about this in general, or in other special cases than those already described?

Bibliography

[ACM98] *Communications of the ACM*, Vol. 41, No. 7, July 1998.

[AP98] R. J. Anderson and F. A. P. Petitcolas, "On The Limits of Steganography", *IEEE Journal on Selected Areas in Communications, Special Issue on Copyright & Privacy Protection*, vol 16 no. 4, pp. 474-481, May 1998.

[BCEKZ00] A. Barg, G. Cohen, S. Encheva, G. Kabatiansky and G. Zémor, *A hypergraph approach to the identifying parent property: the case of multiple parents*, DIMACS Technical Report 2000-20, August 2000.

[BF99] D. Boneh and M. Franklin, "An Efficient Public Key Traitor Tracing Scheme", *Proceedings of Crypto '99*, Springer-Verlag, pp. 338-353, 1999.

[BMP85] G. R. Blakley, C. Meadows and G. B. Purdy, "Fingerprinting Long Forgiving Messages", *Proceedings of Crypto '85*, Springer-Verlag, pp. 180-189, 1985.

[BS95] D. Boneh and J. Shaw, "Collusion-Secure Fingerprinting for Digital Data", *Advances in Cryptology: Proceedings of Crypto '95*, Springer-Verlag, pp. 452-465, 1995.

[BS98] D. Boneh and J.Shaw, "Collusion-Secure Fingerprinting for Digital Data", *IEEE Trans. Inform. Theory*, vol IT-44, pp. 1897-1905, Sep. 1998.

[BSCG] Crypto-Gram, a free monthly e-mail newsletter about computer security and cryptography. Availible from:
http://www.counterpane.com/crypto-gram.html

[CFN94] B. Chor, A. Fiat and M. Naor, "Tracing Traitors", *Proceedings of Crypto '94*, Springer-Verlag, pp. 257-270, 1994.

[CFNP00] B. Chor, A. Fiat, M. Naor and B. Pinkas, "Tracing Traitors", *IEEE Trans. Inform. Theory*, vol IT-46, pp. 893-910, May. 2000.

[HK99] F. Hartung and M. Kutter, “Multimedia Watermarking Techniques”, *Proceedings of the IEEE*, vol. 87, No. 7, pp. 1079 - 1107, Jul. 1999.

[HvLLT98] H. D. L. Hollman, J. H. van Lint, J-P. Linnartz and L. M. G. M. Tolhuizen, “On Codes with the Identifiable Parent Property”, *Journal of Combinatorial Theory*, pp. 121-133, Series 82, 1998.

[JME98] J. M. Ettinger, “Steganalysis and Game Equilibria”, *Proceedings of the Second International Workshop on Information Hiding*, Springer-Verlag, pp. 319-328, 1998.

[JL98] J. Löfvenberg, “Random codes for digital fingerprinting”, *Proceedings 1998 IEEE International Symposium on Information Theory*, p. 80, 1998.

[JL99] J. Löfvenberg, *Random Codes for Digital Fingerprinting*, Linköping Studies in Science and Technology, Thesis No. 749, 1999.

[KP00] S. Katzenbeisser and F. A. P. Petitcolas (editors), *Information Hiding Techniques for Steganography and Digital Watermarking*, Norwood: Artech House, 2000, ISBN 1-58053-035-4.

[LLW97] T. Lindkvist, J. Löfvenberg and N. Wiberg, *Fingerprinting of Digital Information—Introduction and some Preliminary Results*, Report LiTH-ISY-R-1985, ISSN 1400-3902. Availible at: http://www.it.isy.liu.se/research

[MWS77] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam: North-Holland, 1977.

[NNV77] N. N. Vorob’ev, *Game Theory Lectures for Economists and System Scientists*, Springer Verlag, New York 1977, ISBN 0-387-90238-4.

[NSS99] D. Naccache, A. Shamir and J. P. Stern, “How To Copyright a Function ?”, *Proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography*, Springer-Verlag, pp. 188-196, 1999.

[NW83] N. Wagner, “Fingerprinting”, *Proceedings of the 1983 Symposium on Security and Privacy*, pp. 18-22, April 1983.

[PS96] B. Pfitzmann and M. Schunter, “Asymmetric Fingerprinting”, *Advances in Cryptology – EUROCRYPT ‘96*, Springer-Verlag, pp. 85-95, 1996.

[PS99] B. Pfitzmann and A-R. Sadeghi, “Coin-Based Anonymous Fingerprinting”, *Advances in Cryptology – EUROCRYPT ‘99*, Springer-Verlag, pp. 150-164, 1999.

[PW97] B. Pfitzmann and M. Waidner, “Anonymous Fingerprinting”, *Advances in Cryptology – EUROCRYPT ‘97*, Springer-Verlag, pp. 88-102, 1997.

[SR92] S. Roman, *Coding and Information Theory*, Springer-Verlag, New York 1992, ISBN 0-387-97812-7.

[SSW00] J. N. Staddon, D. R. Stinson and R. Wei, "Combinatorial Properties of Frameproof and Traceability Codes", *IEEE Trans. Inform. Theory*, vol IT-47, pp. 1042-1049, March 2001.

[SvTW00] D. R. Stinson, T. van Trung and R. Wei, "Secure Frameproof Codes, Key Distribution Patterns, Group Testing Algorithms and Related Structures", *Journal of Statistical Planning and Inference*, 86 (2000), pp. 595-617.

[SW98] D. R. Stinson and R. Wei, "Combinatorial properties and constructions of traceability schemes and frameproof codes", *SIAM Journal of Discrete Mathematics* 11, pp. 41-53, 1998.

[TL97] T. Lindkvist, *A Short Note about Binary Lineary Codes for Fingerprints*, Report LiTH-ISY-EX-R-1986, ISSN 1400-3902, Availible at: http://www.it.isy.liu.se/research

[TS84] T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications", *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 776-780, Sep. 1984.

Appendix A

Lemmas on Decoding

Let the code $C = \{c_0, \ldots, c_{|C|-1}\}$ be a subset of the finite set $X = \{x_0, \ldots, x_{|X|-1}\}$. X is the set of possible received elements in a communication system and we want to decode any received elements to elements in C. The set of mappings we will choose from when we decide how to decode is the set of memoryless random decodings, $\mathcal{A}$. We will say that two decodings a and b are equivalent if the probability of decoding x to c using a is equal to the probability of decoding x to c using b, for all $x \in X$ and all $c \in C$.

With this view on equivalence among decodings, the general way of describing decodings is to list the probabilities of decoding x_i to c_j for all $i = 0, 1, \ldots, |X| - 1$ and $j = 0, 1, \ldots, |C| - 1$. We will denote each such probability by $\alpha_{i,j}$, and we get that the elements $\alpha_{i,j}$ fulfil the property

$$\sum_{j=0}^{|C|-1} \alpha_{i,j} = 1, \; i = 0, 1, \ldots, |X| - 1. \qquad \text{(A:1)}$$

The collection of such elements $\alpha_{i,j}$ describe a random decoding $A \in \mathcal{A}$.

One way of decoding is deterministically. In this case the decoding is just a deterministic mapping from the indices of the elements in X to indices of the elements in C, $D_i:\{0, \ldots, |X|-1\} \rightarrow \{0, \ldots, |C|-1\}$. We will denote the set of such mappings by $\mathcal{D} = \{D_0, D_1, \ldots, D_{|\mathcal{D}|-1}\}$, where $|\mathcal{D}| = |C|^{|X|}$.

Another way of decoding is by mixed decoding. In mixed decoding, when an element is received, a deterministic decoding is chosen randomly from $\mathcal{D}$. The probabilities of choosing the different elements in $\mathcal{D}$ defines the mixed decoding. We will denote the set of mixed decodings by $\mathcal{B}$, and represent any element $B \in \mathcal{B}$ with the probability vector $(\beta_0, \beta_1, \ldots, \beta_{|\mathcal{D}|-1})$, where each $\beta_a \in [0, 1]$ and $\sum_{a=0}^{|\mathcal{D}|-1} \beta_a = 1$.

Lemma 1.1: $\mathcal{D} \subseteq \mathcal{B} \subseteq \mathcal{A}$

Proof: We first prove that $\mathcal{D} \subseteq \mathcal{B}$. Choose any $D_i \in \mathcal{D}$. By choosing $B \in \mathcal{B}$ such that $\beta_i = 1$ and $\beta_j = 0$, $\forall j \neq i$ we get a decoding that is equivalent to D_i, meaning that we can identify B with D_i so that $D_i \in \mathcal{B}$, and since D_i was arbitrary this is true for all elements in $\mathcal{D}$, and thus $\mathcal{D} \subseteq \mathcal{B}$.

We now go on proving that $\mathcal{B} \subseteq \mathcal{A}$. Choose any $B = \mathcal{B}$, represented by the probability vector $(\beta_0, \beta_1, \ldots, \beta_{|\mathcal{D}|-1})$. With this B the probability of decoding a received x_i to the codeword c_j is

$$\sum_{D_a \in \mathcal{D}: D_a(i) = j} \beta_a .$$

By choosing $A \in \mathcal{A}$ such that

$$\alpha_{i,j} = \sum_{D_a \in \mathcal{D}: D_a(i) = j} \beta_a \qquad \text{(A:2)}$$

we get a decoding that is equivalent to *B*, meaning that we can identify *A* with *B*, so that $B \in \mathcal{A}$, and since *B* is arbitrary this is true for all elements in $\mathcal{B}$, and thus $\mathcal{B} \subseteq \mathcal{A}$.

❑

Lemma 1.2: $\mathcal{A} \subseteq \mathcal{B}$

Choose any $A \in \mathcal{A}$, represented by the probabilities $\alpha_{i,j}$, $i = 1, \ldots, |X| - 1$, $j = 1, \ldots, |C| - 1$. We will show that *B* with

$$\beta_a = \prod_{k=0}^{|X|-1} \alpha_{k, D_a(k)} \tag{A:3}$$

is equivalent to *A*. We will show this by proving that the $A' \in \mathcal{A}$ equivalent to *B* that can be found using equation (A:2), is the same as the arbitrarily chosen *A*.

Inserting the β-values of *B* from (A:3) into (A:2) we get:

$$\alpha'_{i,j} = \sum_{D_a \in \mathcal{D}: D_a(i) = j} \prod_{k=0}^{|X|-1} \alpha_{k, D_a(k)}$$

$$= \sum_{D_a \in \mathcal{D}: D_a(i) = j} \left(\alpha_{i, D_a(i)} \prod_{k=0, k \neq i}^{|X|-1} \alpha_{k, D_a(k)} \right)$$

$$= \sum_{D_a \in \mathcal{D}: D_a(i) = j} \left(\alpha_{i,j} \prod_{k=0, k \neq i}^{|X|-1} \alpha_{k, D_a(k)} \right) =$$

$$= \alpha_{i,j} \sum_{D_a \in \mathcal{D}: D_a(i) = j} \prod_{k=0, k \neq i}^{|X|-1} \alpha_{k, D_a(k)} . \qquad \text{(A:4)}$$

Since $\mathcal{D}$ contains all possible mappings $D:\{0, \ldots, |X|-1\} \rightarrow \{0, \ldots, |C|-1\}$, the sum in (A:4) is taken over *all* mappings $D:\{0, \ldots, |X|-1\} \rightarrow \{0, \ldots, |C|-1\}$ such that $D(i) = j$. This means that the sequences $[D_a(k)]_{k=0, k \neq i}^{|X|-1}$, taken over $D_a \in \mathcal{D}: D_a(i) = j$ is exactly *all* sequences of length $|X|-1$ with each element chosen from $\{0, \ldots, |C|-1\}$. This means that (A:4) can be further transformed as follows.

$$\alpha_{i,j} \sum_{D_a \in \mathcal{D}: D_a(i) = j} \prod_{k=0, k \neq i}^{|X|-1} \alpha_{k, D_a(k)}$$

$$= \alpha_{i,j} \prod_{k=0, k \neq i}^{|X|-1} \sum_{t=0}^{|C|-1} \alpha_{k,t} = \alpha_{i,j} \qquad \text{(A:5)}$$

We get the last equality in (A:5) by using relation (A:1), $\sum_{j=0}^{|C|-1} \alpha_{i,j} = 1$. (A:5) means that $\alpha'_{i,j} = \alpha_{i,j}$, and thus that $A' = A$ which is what we need to prove.

❑

The only thing left to show is that the vector $(\beta_0, \beta_1, \ldots, \beta_{|\mathcal{D}|-1})$ is actually a probability vector, and thus describes a mixed decoding $B \in \mathcal{B}$.

Lemma 1.3: Choosing β_a as in expression (A:3) yields β_a such that $\beta_a \in [0, 1]$ and $\sum_{a=0}^{|\mathcal{D}|-1} \beta_a = 1$.

Proof: $\beta_a \in [0, 1]$ since it is the product of probabilities $\alpha_{i,j}$. We now have to prove that $\sum_{a=0}^{|\mathcal{D}|-1} \beta_a = 1$.

$$\sum_{a=0}^{|\mathcal{D}|-1} \beta_a = \sum_{D_a \in \mathcal{D}} \beta_a = \sum_{j=0}^{|C|-1} \left(\sum_{D_a \in \mathcal{D}: D_a(i) = j} \beta_a \right) = \sum_{j=0}^{|C|-1} \alpha_{i,j} = 1,$$

where the last two equalities are due to equation (A:2) and (A:1) respectively.

❑

Lemmas 1.2 and 1.3 together show that $\mathcal{A} = \mathcal{B}$.

Appendix B

Tables of Optimal and Best Known Codes

B.1 Tables of Optimal Codes

In the tables the same tags as in section 8.5 have been used to classify the codes.

B.1.1 Codes of size 3

n	Code matrix	$1 - \mu(\Gamma)$
3	0 0 0 0 1 1 1 0 1	0.1250 IM

Table B.1: *Optimal codes of size 3, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
4	0 0 0 0 0 0 1 1 1 1 0 1	0.0833 Ix
5	0 0 0 0 0 0 0 1 1 1 1 1 0 0 1	0.0521 Ix
6	0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1	0.0313 IM
7	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1	0.0208 Ix
8	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1	0.0130 Ix
9	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1	0.0078 IM
10	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1	0.0052 Ix

Table B.1: *Optimal codes of size 3, with respect to* $\mu(\Gamma)$.

B.1.2 Codes of size 4

n	Code matrix	$1-\mu(\Gamma)$
3	0 0 0 1 1 0 1 0 1 0 1 1	0.3000 Ix
4	0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1	0.1500 IM
5	0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0 0 0 1	0.1125 Ix
6	0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0	0.0750 M2
7	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0	0.0500 Mx

Table B.2: *Optimal codes of size 4, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
8	0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0	0.0313 Mx
9	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0	0.0188 M2
10	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0	0.0125 Mx

Table B.2: *Optimal codes of size 4, with respect to* $\mu(\Gamma)$.

B.1.3 Codes of size 5

n	Code matrix	$1-\mu(\Gamma)$
3	0 0 0 0 0 1 0 1 1 1 0 1 1 1 1	0.8000

Table B.3: *Optimal codes of size 5, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
4	0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1	0.3000 Ix
5	0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 1 1 0 0 0 1	0.1667 IM
6	0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 0 0 0 1 1	0.1333 Ix
7	0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0	0.0969

Table B.3: *Optimal codes of size 5, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1 - \mu(\Gamma)$
8	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0	0.0625
9	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0	0.0453

Table B.3: *Optimal codes of size 5, with respect to* $\mu(\Gamma)$.

B.1.4 Codes of size 6

n	Code matrix	$1 - \mu(\Gamma)$
4	0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 1 0 1 1 1 0 1	0.6786

Table B.4: *Optimal codes of size 6, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
5	0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0	0.2857 Ix
6	1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1	0.1786 IM
7	0 0 0 0 0 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 1	0.1488 Ix
8	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 1 0 1 1 0 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 0 1 1	0.1071

Table B.4: *Optimal codes of size 6, with respect to* $\mu(\Gamma)$.

B.1.5 Codes of size 7

n	Code matrix	$1 - \mu(\Gamma)$
4	0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1	0.8393
5	0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 0 0 1 0 1	0.5982
6	0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1	0.2768 Ix

Table B.5: *Optimal codes of size 7, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
7	0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1	0.1875 IM

Table B.5: *Optimal codes of size 7, with respect to* $\mu(\Gamma)$.

2.1.6 Codes of size 8

n	Code matrix	$1-\mu(\Gamma)$
4	0 0 0 0 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 1 0	0.8750

Table B.6: *Optimal codes of size 8, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1 - \mu(\Gamma)$
5	0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0	0.7431
6	0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1	0.5417

Table B.6: *Optimal codes of size 8, with respect to* $\mu(\Gamma)$.

B.1.7 Codes of size 9

n	Code matrix	$1-\mu(\Gamma)$
4	0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1	0.8889
5	0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 1 0 0 0	0.7833

Table B.7: *Optimal codes of size 9, with respect to* $\mu(\Gamma)$.

B.1.8 Codes of size 10

n	Code matrix	$1-\mu(\Gamma)$
4	All codes	1
5	$\begin{matrix}0&0&0&0&0\\1&1&0&1&0\\0&1&0&0&0\\0&1&1&1&0\\1&1&0&0&0\\0&1&0&1&1\\1&1&0&1&1\\0&1&0&0&1\\1&1&0&0&1\\0&1&0&1&0\end{matrix}$	0.8045

Table B.8: *Optimal codes of size 10, with respect to* $\mu(\Gamma)$.

B.2 Tables of Best Known Codes

In the tables the same tags as in section 8.5 have been used to classify the codes.

B.2.1 Codes of size 5

n	Code matrix	$1-\mu(\Gamma)$
10	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 1 1 0 1 0 0 1 1 1 0	0.0312 CW

Table B.9: *Best found codes of size 5, with respect to* $\mu(\Gamma)$.

B.2.2 Codes of size 6

n	Code matrix	$1-\mu(\Gamma)$
9	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 1 1 1 0 1 1 0 1 0	0.0714 Mx

Table B.10: *Best found codes of size 6, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
10	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0 0	0.0446 M2

Table B.10: *Best found codes of size 6, with respect to* $\mu(\Gamma)$.

B.2.3 Codes of size 7

n	Code matrix	$1-\mu(\Gamma)$
8	0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1	0.1607 Ix
9	0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1	0.1362 Ix

Table B.11: *Best found codes of size 7, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
10	0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1	0.1138 Ix

Table B.11: *Best found codes of size 7, with respect to* $\mu(\Gamma)$.

B.2.4 Codes of size 8

n	Code matrix	$1-\mu(\Gamma)$
7	0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 1	0.2708 Ix

Table B.12: *Best found codes of size 8, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
8	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1	0.1944 IM
9	0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1	0.1701 Ix
10	0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1	0.1476 Ix

Table B.12: *Best found codes of size 8, with respect to* $\mu(\Gamma)$.

B.2.5 Codes of size 9

n	Code matrix	$1-\mu(\Gamma)$
6	0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 1 1 1 0 0	0.6525
7	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0	0.5000

Table B.13: *Best found codes of size 9, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
8	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1	0.2667 Ix
9	1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	0.2000 IM
10	1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1	0.1778 Ix

Table B.13: *Best found codes of size 9, with respect to* $\mu(\Gamma)$.

B.2.6 Codes of size 10

n	Code matrix	$1-\mu(\Gamma)$
6	0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1	0.6833
7	0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 1 0 1 0	0.6273

Table B.14: *Best found codes of size 10, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
8	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0	0.4682
9	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	0.2636 Ix

Table B.14: *Best found codes of size 10, with respect to* $\mu(\Gamma)$.

n	Code matrix	$1-\mu(\Gamma)$
10	1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1	0.2045 IM

Table B.14: *Best found codes of size 10, with respect to* $\mu(\Gamma)$.