
Collusion Secure Fingerprint Watermarking

Koalitionssichere Fingerprint-Wasserzeichen-Verfahren

Zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation von Marcel Schäfer (Diplom-Mathematiker) aus Solingen
Januar 2016 – Darmstadt – D 17



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Sicherheit in der Informationstechnik
Fraunhofer-Institut für sichere
Informationstechnologie SIT
CASED – Center for Advanced Security
Research Darmstadt

Collusion Secure Fingerprint Watermarking
Koalitionssichere Fingerprint-Wasserzeichen-Verfahren

Genehmigte Dissertation von Marcel Schäfer (Diplom-Mathematiker) aus Solingen

1. Gutachten: Prof. Dr. Michael Waidner
2. Gutachten: Prof. Dr. Rüdiger Grimm
3. Gutachten: Dr. Martin Steinebach

Tag der Einreichung: 21.08.2015

Tag der Prüfung: 12.01.2016

Darmstadt – D 17

Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den January 15, 2016

(M. Schäfer)

Abstract

Identifying perpetrators via watermarking technology has proven of value in media copyright infringements. To enable tracing back unauthorizedly re-distributed media copies that were manipulated by a collusion attack, collusion secure fingerprinting codes are embedded into the copies via watermarking technology. Fingerprinting codes are mathematical codes designed to resist collusion attacks by means of probabilistically generated codewords and suitable tracing algorithms.

However, embedding fingerprinting codes as watermark messages is challenging. This is because of its enormous code length in realistic parameter settings such as small error rates and resistance against large collusions, while the watermarking payload provided by the media copies is very limited. Therefore, to first enable the use of fingerprinting codes in practice, a suitable compromise regarding the fingerprinting parameters must be identified. As the different media fields come with varying requirements on the fingerprinting codes, it is impossible to rely on one specific fingerprinting scheme. Instead, for each of the different media application scenarios separately, the suitable parameter setting is to be identified. With it, conceptuation and design of specifically tailored fingerprinting solutions for the different requirements is first possible.

This thesis takes on these challenges to design and optimize various fingerprinting codes each matching a certain application scenario and thereby finally enabling the application of fingerprinting codes for those scenarios in need of collusion security. Hence, we identify a number of application scenarios that are potentially subject to collusion attacks. We give definitions of the different types of fingerprinting schemes, its parameters and properties as well as collusion attacks. The general structure of modern fingerprinting schemes is derived and explained with the example of the Tardos codes. With respect to these definitions, the different application scenarios are analyzed in order to identify the individual requirements. With it we are able to postulate what types of fingerprinting schemes with which parameter settings suffice these requirements. To that effect, various fingerprinting codes are proposed. We developed fingerprinting codes designed to resist very small collusions, while providing shortest code lengths at comparably low error rates. Moreover, different approaches are given, optimizing the generation of the fingerprinting codes with respect to practical applications and appropriate parameter settings. To that, we also developed the corresponding tracing strategies, imposing improvements regarding error rates, code length, complexity and that are independent of the attacks. Besides, we also showed how fingerprinting codes can be realized in hashtable database scenarios thus enabling collusion security also for this scenario. The proposed approaches all have their specific advantages, for which each is the most appropriate for a specific application scenario and use case. This allows to apply fingerprinting codes in practice and provides collusion security for various media scenarios.

Abstract (Deutschsprachige Version)

Das Identifizieren von Verrätern mittels Wasserzeichen-Technologie ist eine bewährte Methode, um Urheberrechtsverletzungen in Multimedia zu verfolgen. Um auch diejenigen der unbefugt weiterverteilten Kopien zurück verfolgen zu können, welche einem Koalitionsangriff unterliegen, werden koalitionsresistente Fingerprinting-Codes mittels Wasserzeichen-Technologie in die Kopien eingebettet. Fingerprinting-Codes sind mathematische Codes mit der Eigenschaft auch Koalitionsangriffe überstehen zu können. Sie bestehen aus wahrscheinlichkeitstheoretisch generierten Codewörtern und entsprechend geeigneten Tracing-Algorithmen. Das Einbetten von Fingerprinting-Codes als Wasserzeichen-Nachrichten stellt jedoch eine große Herausforderung dar. Der Grund dafür ist die enorme Codelänge der Fingerprints bei realistischen Parametereinstellungen, etwa eine geringe Fehlerrate bei gleichzeitiger Resistenz gegen große Koalitionen, verglichen mit der stark begrenzten Wasserzeichen-Nutzlast der Multimedia-Trägerdateien. Daher muss erst ein geeigneter Kompromiss bezüglich der Fingerprinting-Parameter gefunden werden, um die Verwendung von Fingerprinting-Codes in der Praxis überhaupt zu gewährleisten.

Da die verschiedenen Multimedia-Bereiche unterschiedliche Anforderungen an die Fingerprinting-Codes stellen, ist es nicht möglich, nur auf ein einziges Fingerprinting-System zurück zu greifen. Stattdessen muss erst für jedes der verschiedenen Multimedia-Bereiche eine geeignete Parametereinstellung gefunden werden. Damit erst sind Konzeptuierung und Gestaltung von individuell zugeschnittenen Fingerprinting-Lösungen für die unterschiedlichen Anforderungen möglich.

Diese Arbeit beschäftigt sich mit diesen Herausforderungen, um diverse Fingerprinting-Codes zu entwickeln und zu optimieren. Diese sollen die unterschiedlichen Anwendungsfälle abdecken und somit erst die Anwendung von Fingerprinting-Codes in ausgesuchten Szenarien ermöglichen. Dafür identifizieren wir eine Reihe von Anwendungsszenarien, die potentielle Ziele von Koalitionsangriffen sind. Wir geben Definitionen, Parameterbeschreibungen und Eigenschaften der verschiedenen Fingerprinting-Schemata sowie Koalitionsangriffe. Das allgemeine Schema moderner Fingerprinting-Systeme wird anhand der Tardos Codes hergeleitet und erläutert. Im Hinblick auf diese Definitionen werden die verschiedenen Anwendungsszenarien analysiert, um die individuellen Anforderungen zu identifizieren. Damit sind wir in der Lage zu postulieren, welche Fingerprinting-Systeme mit welchen Parametereinstellungen diesen Anforderungen genügen. Zu diesem Zweck werden verschiedene Fingerprinting-Codes vorgestellt. Wir entwickeln Fingerprint-Codes mit signifikant kurzen Codelängen bei vergleichsweise niedrigen Fehlerraten, um speziell gegen sehr kleine Koalitionen resistent zu sein. Darüber hinaus werden verschiedene Ansätze vorgestellt, welche die Generierung der Fingerprinting-Codes im Hinblick auf praktische Anwendungen und entsprechende Parametereinstellungen optimieren. Passend dazu entwickeln wir auch die entsprechenden Tracing-Algorithmen, die Verbesserungen in Bezug auf Fehlerquoten, Codelänge und Komplexität herbei führen und auch solche, die unabhängig von den geführten Koalitionsangriffen sind. Außerdem zeigen wir, wie Fingerprinting-Codes in Hashtable-Datenbank Szenarien realisiert werden können und somit auch für dieses Gebiet Koalitionssicherheit ermöglichen. Jeder der vorgestellten Ansätze für sich hat gewisse Vorteile, so dass jeder für ein bestimmtes Szenario und einen bestimmten Anwendungsfall am besten geeignet ist. Dies ermöglicht die Verwendung von Fingerprinting-Codes in der Praxis und bietet Koalitionssicherheit für diverse Szenarien.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	History	2
1.3	Goals	3
1.4	Course of action	4
2	Security mechanisms for media	6
2.1	Objectives for watermarking and fingerprinting	6
2.2	Security components for media	7
2.3	Authenticity	8
2.4	Transactional watermarking	9
2.5	Collusion attacks	12
2.6	Security components against collusion attacks	13
2.7	Summary	14
3	Application scenarios for fingerprinting codes	15
3.1	General classification of the application scenarios for fingerprinting codes	15
3.2	Selection criteria for the considered scenarios	16
3.3	Promotion and pre-release application scenario	17
3.4	Online shop application scenario	18
3.5	Digital cinema application scenario	20
3.6	Boxed video game application scenario	21
3.7	Database/hashtable application scenario	24
4	Collusion secure fingerprinting codes: Concepts and properties	26
4.1	The general fingerprinting scheme	26
4.2	Types of fingerprinting schemes	27
4.3	Fingerprinting parameters	29
4.4	Marking assumption	33
4.5	Symmetric Tardos scheme	34
4.5.1	Tardos fingerprint generation	34
4.5.2	Fingerprint generation with discrete bias distribution	35
4.5.3	Symmetric Tardos tracing algorithm	37
4.6	Sophisticated fingerprinting attacks	39
4.6.1	Stateless attack strategies	39
4.6.2	Stateful attack strategies	41
4.7	Summary	42
5	Requirement analysis of fingerprinting codes in the considered application scenarios	44
5.1	Fingerprinting codes for promotional and pre-release purposes	44
5.2	Fingerprinting codes for online shops	47
5.3	Fingerprinting codes for digital cinema	49
5.4	Fingerprinting codes for video games	50
5.5	Fingerprinting codes for databases/hashtables	51
5.6	Summary	52

6	Short fingerprinting codes against small collusions	56
6.1	2-Secure fingerprinting	57
6.1.1	Preliminaries	57
6.1.2	Construction	58
6.1.3	Evaluation	61
6.2	3-Secure Fingerprinting	62
6.2.1	Preliminaries	62
6.2.2	Construction	64
6.2.3	Evaluation	67
6.3	Summary	69
7	Optimized fingerprinting codes against medium sized collusions	70
7.1	Efficient tracing by filtering suspects: Ranking search	71
7.1.1	Construction	71
7.1.2	Explanation of the algorithm	73
7.1.3	Evaluation	75
7.2	Optimized fingerprint generation for the continuous distribution	77
7.2.1	Preliminaries	77
7.2.2	Construction	78
7.2.3	Evaluation	80
7.3	Adapted fingerprint generation for discrete bias distributions	81
7.3.1	Construction	81
7.3.2	Evaluation	83
7.4	Dynamic threshold computation for the interleaving score function	85
7.4.1	Construction	85
7.4.2	Evaluation	87
7.5	Summary	89
8	Short and efficient fingerprinting codes	91
8.1	Efficient joint decoding tracing algorithm	92
8.1.1	Preliminaries	92
8.1.2	Construction	93
8.1.3	Evaluation	95
8.2	Optimized fingerprint generation for continuous distributions II	97
8.2.1	Construction	97
8.2.2	Evaluation	98
8.3	Universal threshold calculation for large collusions	100
8.3.1	Construction	100
8.3.2	Evaluation	103
8.4	Summary	106
9	Fingerprinting codes for database tracing	108
9.1	Construction of a collusion secure hash table	108
9.2	Detection and tracing	109
9.3	Attack models	110
9.4	Evaluation and discussion	111
9.5	Summary	113

10 Application scenarios supporting fingerprinting codes	114
10.1 Fingerprinting codes for promotional application scenarios	114
10.2 Fingerprinting codes for online shops	115
10.3 Fingerprinting solution for digital cinema	117
10.4 Fingerprinting solution for boxed video games	119
10.5 Fingerprinting solution for hash table databases	121
11 Open challenges and opportunities	122
11.1 Toolbox containing all recent and state of the art fingerprinting codes	122
11.2 Higher alphabets for watermarking basis and fingerprinting codes	122
11.3 Sophisticated collaboration of watermarking and fingerprinting technologies . . .	124
11.4 Additional application scenarios for fingerprinting codes	124
11.5 Combination of other security mechanisms and fingerprinting codes	125
12 Summary and conclusions for future work	127
A Optimal parameter selection for the symmetric Tardos scheme	139
B Regarding the stateful attack strategies	143
C Regarding the 2-secure fingerprinting scheme	146
C.1 The false positive error	146
C.2 The false negative error	150
C.3 Error probabilities for collusions larger than two	151
D Regarding the 3-secure fingerprinting scheme	153
E Regarding the fingerprint matrix generation	155
F Regarding the discrete distribution	158
F.1 Continuing explanations	158
F.2 Continuing evaluations	159
G Regarding the dynamic threshold	162
H Regarding the joint decoding approach	165
I Regarding the correlative column generation II	167
J Regarding the universal threshold calculation	173
J.1 Derivations of the parameter calculations	173
J.2 Test setup for the fitting of mixture models and backup threshold if the fitting fails	176
J.3 Tests for the minority and majority vote attack	176
J.4 Combination of correlated matrix generation and GMM	179

List of Figures

1	Boxed video game components to be watermarked	23
2	Tardos probability density function and score functions g_1 and g_0 , according to [88]	36
3	Comparison of the code length parameter for continuous and discrete distributions	37
4	Normalized distribution of the accusation scores	38
5	Different watermarking layers in the life-cycle of a video game	51
6	The probability density function f'	65
7	Majority tracing strategy	66
8	Minority tracing strategy	68
9	Example for creation of the ranking vector V	74
10	Example for creation of the ranking matrix R	74
11	Histogramm of the accusation sums at Tardos' and our matrix generation	79
12	Comparison of results for discrete distribution [65] and our adaption	83
13	Different attacks on the dynamic threshold calculation [88]	87
14	Example of a two-component mixture model consisting of two normal distributions.	101
15	Hashtable Individualization Scheme	109
16	Leakage detection without access to the hash database	111
17	Minimal code length parameter d_m for $\varepsilon_1 = \varepsilon_2$ and $\varepsilon_1 = 10^{-8}, \varepsilon_2 = 1/2$	141
18	Example for a disadvantageous constellation of 2-secure fingerprints, for which an innocent-fingerprint looks very similar to the manipulated fingerprint y	146
19	Box plot of the error rates for minority vote minimal mutual information attack	155
20	Squared distance of the probability values p_i from the center 0.5 in comparison to the error rate	156
21	Comparison of false positive for discrete distributions	159
22	Empirically measured false positive rate and approximated Gaussian model	160
23	Error rates for symmetric and interleaving score function	163
24	FP error for symmetric and interleaving score function	163
25	Percentage of colluder fingerprints caught in different attack scenarios	165
26	Analysis of the correlation coefficients: arithmetic mean and empirical variance	170
27	Analysis of the correlation coefficients: skewness and excess	171
28	Interleaving attacks on correlated fingerprinting codes: error rates	171
29	Interleaving attacks on correlated fingerprinting codes: standard deviation	172

List of Tables

1	Attacks that minimize the mutual information between y and colluders	41
2	Classification of the parameters payload, number of users and maximal expected collusion size in the promotional application scenario	47
3	Classification of the parameters payload M , number of users n and maximal expected collusion size c_0 in the Online shop application scenario	48
4	Identified requirements on the fingerprinting codes for the different application scenarios	53
5	2-secure fingerprinting approach: Comparison of code length and errors to [79] .	61
6	3-secure fingerprinting approach: Probability of a FP	68
7	3-secure fingerprinting approach: Comparison to [82]	68
8	Ranking search approach: Results for $c_0 = 3$	75
9	Ranking search: Results for $c_0 = 6$ after the first step	76
10	Adapted fingerprint generation for discrete distributions: Comparison to random generation	83
11	Dynamic threshold approach and adapted fingerprint generation: Comparison of code lengths	88
12	Joint tracing approach: Comparison of code lengths with single tracing approach of [121]	95
13	Joint tracing approach: Results for different attack strategies	96
14	Correlated fingerprint generation: Comparison to uncorrelated generated fingerprints	98
15	Universal threshold calculation: Comparison of different score functions in various settings	103
16	Hashtable fingerprinting: Comparison of different code lengths and attacks	112
17	Fingerprinting solutions for the promotional application scenario	115
18	Fingerprinting codes for the online shop application scenario	116
19	Fingerprinting codes for the digital cinema application scenario	118
20	Fingerprinting codes for the boxed video game application scenario	120
21	Optimal parameter settings according to [64]	141
22	2-secure fingerprinting: False negative error in different settings II	150
23	2-secure fingerprinting: False negative error in different settings I	151
24	3-secure fingerprinting approach: Comparison of different attacks	153
25	Fingerprint generation: Parameter selection according to [64]	156
26	Adapted fingerprint generation: Correlation between error rate and statistical properties of fingerprint matrix X	157
27	Joint tracing approach: Comparison for small c_0	166
28	Joint tracing approach: Results for $c_0 = 12$	166
29	Universal threshold calculation: Comparison for threshold calculations and different score functions after minority vote attack	177
30	Universal threshold calculation: Comparison for threshold calculations and different score functions after majority vote attack	178
31	Universal threshold calculation and correlated fingerprint generation: Comparison to uncorrelated fingerprint generation and threshold calculation according to [121]	179

1 Introduction

This thesis concerns itself with collusion secure fingerprinting codes for digital watermarking applications. In the process new fingerprinting schemes are presented that allow sophisticated security models for watermarking schemes. The fingerprinting algorithms are special tailored for different watermarking applications and varying requirements that are caused by diverse media application scenarios.

1.1 Motivation

Digital transaction watermarking is a technique to protect copyright after the loss of control of digital media once they are published, i.e. to counter unauthorized copying and redistributing of media over the Internet. Using digital watermarking algorithms one is able to imperceptibly hide information in digital media. In a digital distribution scenario, watermarking is used to embed individual information into digital copies of the same content in order to distinguish between these copies. Referring to the transaction ID of a customer who purchases a digital copy of content, this information is called transaction watermark message. It allows to clearly associate a customer to his purchased copy. Thereby a method is provided to trace back unauthorizedly redistributed copies to the responsible customer(s).

First approaches to protect digital media via transaction watermarking methods are ascribed to the middle of the 1990s, though broad industrial interest came up in the first years of the 21st century. Nowadays, the mechanisms and techniques of digital watermarking achieve a security level that makes it hard for attackers to destroy or manipulate the watermark without strong degradation of the cover media. However, by means of collusion attacks potential perpetrators are given a method to create a high quality media copy for which the watermark is massively counterfeited. This is done by comparing two or more copies of the same content in order to detect the differences induced by the individual transaction watermark messages. Manipulating only at these positions a copy of the content can be created that provides equal quality but the transaction watermark is strongly tampered. For this reason, collusion attacks still form a huge issue for the industry and protection against collusion attacks is of denotative interest.

A solution against collusion attacks is provided by collusion secure fingerprinting codes or simply fingerprinting codes. These are mathematical codes, for simplicity denoted as fingerprints, that can be embedded as transaction watermarks via transaction watermarking algorithms in order to correctly identify the traitors of a manipulated watermark even after a collusion attack. Hence, a fingerprint acts as a codeword that is generated to serve as a clear identifier for a user or source. A detailed description of collusion secure fingerprinting codes is given in chapter 4.

Reducing the length of collusion secure fingerprinting codes is the primary concern in related research. This is because the watermarking application that applies fingerprinting codes is strongly limited by the number of information that can be embedded into the cover media file. Multimedia of limited payload such as short audio tracks or images provide payload of only a few bits, whereas proper fingerprinting codes typically demand thousands of bits.

Consequently, for the media industry the application of the existing fingerprinting codes is challenging.

In addition, most research focuses on the information theoretic aspects of collusion secure fingerprinting codes. Ironically, this means optimizing the code lengths for asymptotically large parameters (i.e. number of attackers tends to infinity), whereas the industry requires shortest

code lengths for realistically small parameters. However, as will be shown in this thesis, the asymptotically optimal code does not automatically represent the best choice for the application in practice. Therefore, rethinking the encoders and creators of fingerprinting codes with a more practical perspective, i.e. fingerprinting codes adjusted to the real world requirements, could significantly improve the situation for the media industry.

Moreover, the resistance against collusion attacks, namely the collusion security, for fingerprinting algorithms in general as well as the approaches presented in this thesis are based on probabilistic error rates. This means the output of the fingerprinting tracing algorithm is mathematically proven correct within the limits of the afore chosen error bounds. This implies that there is always a probability that the algorithms fail. Therefore, before court, watermarking and fingerprinting schemes are not accepted as confident proof techniques, but as instruments providing reliable indications. Thus, to serve as proper indications the error rates of fingerprinting codes need to be as small as possible.

However, due to the mathematical nature of fingerprinting codes, reducing the error rates has impact on many other parameters, e.g. the fingerprinting code length. In general, a reduction of the error rates attributes to an increment of code length. Thus, for parameters chosen in an information theoretic sense as well as for parameters matching practical applications, optimizing fingerprinting codes is consistent with improving one parameter while the others do not deteriorate.

In this thesis we present several collusion secure fingerprinting algorithms that can be applied by transaction watermarking applications. With it, a method is provided to protect copyright and detect leakage. However, the presented fingerprinting algorithms are based on the security of the underlying watermarking application. If the watermarking detection algorithm provided fraudulent watermark information, the fingerprinting tracing algorithm that acts based on this fraudulent information would possibly not emit the correct fingerprint. Thus, the sovereignty of the underlying watermarking application is essential. However, the watermarking technology per se lies beyond the scope of this thesis. Watermarking only provides the platform for the application of the proposed fingerprinting algorithms.

Regardless of the strong dependency on the watermarking basis and also of the oppositional requirements regarding practical code lengths and acceptable error probabilities, fingerprinting codes have the potential for successful application in practical scenarios that are prone to collusion attacks.

1.2 History

The term *fingerprinting* in the context of digital data identification already occurred in the 1980s, e.g. *Wagner* [125] and *Blakley et al.* [20]. These early approaches are not based on media watermarking applications and collusion security is achieved simply by enormous fingerprint lengths.

An early approach for *modern* collusion secure fingerprinting codes in watermarking application scenarios is ascribed to *Boneh* and *Shaw* [23]. The significance of this scheme is based on the required code length that is for the first time polynomial in the number of colluding attackers. They also formulated the so called *marking assumption*, that is the attackers only manipulate at the positions where their individually watermarked copies differ from each other, in order to generate a high quality copy that is no longer traceable, see chapter 4. The fingerprinting algorithms proposed in this thesis rely on this assumption regarding all possible attack models.

A milestone approach, the name-giving *Tardos Codes*, was published by *Tardos* in 2003 [114]. In this approach the optimal bound on the code length is proven to be quadratical in the number of colluding attackers, a result that still holds for current schemes. The *Tardos Codes* form the basis of much constructive research until now and is also the basis of most solutions that are presented in this thesis. Therefore a more detailed description of the algorithm and its properties is given in chapter 4.

Important improvements of the *Tardos Codes* are published by *Škorić et al.* in [124] and [121], decoupling the error rates and introducing a symmetric scoring function, respectively. Another approach of value was given by *Blayer* and *Tassa* in [21]. They tightened some inequalities within the proof schemes and thereby achieved further reductions in the code length.

A different approach that provides the shortest code lengths for larger collusion sizes when starting this thesis was published by *Nuida et al.* in 2009 [84]. They used a discrete probability function for the generation of the fingerprints.

We will come back to these approaches in chapters 4, 7 and 8.

For small collusion sizes, the state of the art for the time our work started is marked by *Nuida et al.*, [83], [84] and [82]. We will give a more detailed description of these codes in chapter 6.

Approaches that are strongly oriented to coding theory, i.e. approaches that discuss the information theoretic optimal rate for fingerprinting codes are published among others by *Huang* and *Moulin* [50], [51], [52] and *Amiri* and *Tardos* [7]. These works consider the asymptotic behavior of the fingerprinting codes, i.e. the optimal code while the collusion size tends to infinity. Due to the restrictions inflicted by the multimedia application scenarios, these codes only play a minor role in the process of this thesis.

A detailed carrying out of the related work of fingerprinting codes is shifted to chapter 4.

1.3 Goals

Collusion secure fingerprinting codes are a much discussed research topic but until now found hardly their way into praxis. This is last but not least because the majority of watermarking application providers have only minor knowledge of mathematical fingerprinting codes. The goal of this thesis thus is to describe the different approaches of fingerprinting codes that we developed and to analyze the manifold application scenarios for which existing security vulnerabilities can be shut by means of fingerprinting codes. In order to do so, the different watermarking application scenarios have to be identified and classified according to their especial security and performance requirements. The application scenarios that form the basis are chosen in order to cover the broadest spectrum of types of fingerprinting codes that can be applied in practice.

Based upon this we determine the requirements on the fingerprinting codes and sort them to their potential media application scenario. Given that, it will be illustrated what type of fingerprinting codes finally prove suitable to which of the varying media application scenarios.

Connected to this is the general challenge of reducing the code length of fingerprinting codes. The shorter the code length, the more application scenarios match or the higher will be the achieved security level. However, the fingerprinting parameters, in particular length and security, are strongly connected. More precisely, the code length can be seen as a function depending on the fingerprinting parameters. This means reducing the length of a fingerprinting code can be generalized to optimizing the interaction of the parameters.

Especial attention is spent to the enhancement on the well known *Tardos Codes* [114], which in many scenarios can be seen as the basic tool for the fingerprinting schemes to be applied and already was reason for much research. The uniqueness is reasoned in the targeted applicability in praxis. In contrast to recent information theoretic research by other authors, our focus is to expand or modify the fingerprinting codes according to their actual watermarking application in praxis. Therefore, for this work several fingerprinting codes evolved that are outstanding regarding their simplicity and performance.

In addition, a fundamental new approach is our 2-secure, i.e. resistant against two colluders, fingerprinting code [94], that provides a zero false positive error not only in empirical tests but also mathematically proven. Hereby we intend to expand the applicability also to the legal perspective. An algorithm providing a zero false positive rate is even harder to doubt before court than a false positive probability whatever small it is.

A new application scenario that was recently brought to life within our research group is the scenario of video game watermarking [12]. This scenario by the different nature of video games requires a new kind of fingerprinting approach, as there is no longer a single file to protect by watermarking technology, but sets of files of different media components.

As far as we know, a further novelty is our approach to enlarge the typical application fields for fingerprinting codes by introducing a concept to apply fingerprinting codes in the field of hashtable tracing [14].

Although this work focuses on practical fingerprinting codes for watermarking applications, for lack of space there will not be a deeper introduction into media watermarking algorithms. We will refer to the corresponding algorithm whenever deeper knowledge is required. This work is written with the intention to provide fingerprinting knowledge in a way that is understandable also for readers that are not experts in multimedia and watermarking.

1.4 Course of action

This work describes innovations, optimizations, extensions and adaptations of collusion secure fingerprinting codes with respect to various application scenarios.

In chapter 2 we give an introduction into media security mechanisms. We shortly describe objectives for watermarking and fingerprinting and list other security mechanisms familiar with or sometimes entangled with watermarking or fingerprinting. Besides, we briefly explain authenticity principals and transaction watermarking. Finally, we introduce collusion attacks and methods to counter these.

Chapter 3 identifies the different application scenarios appropriate and in need of collusion secure fingerprinting codes. We investigate a general classification of the scenarios and reason the selection criteria for the scenarios considered in the further proceeding. The rest of this chapter is devoted to describing these scenarios in more detail.

In chapter 4 we present general concepts and properties of collusion secure fingerprinting codes. We define the general fingerprinting scheme and its different types as well as classical fingerprinting parameters. Next we discuss the so called marking assumption and its consequences for the further proceeding of this work. Moreover, we describe in more detail the concept of fingerprinting codes – and with it the state of the art of fingerprint generation and fingerprint tracing – by means of the well known *Tardos* codes. The chapter closes posting various sophisticated collusion attacks.

In chapter 5 we sum up the findings of chapter 2 through 4 and analyze the requirements regarding new approaches or modifications, in order to allow implementation of fingerprinting codes. We determine which fingerprinting scheme to apply at which application scenario.

The chapters 6 through 8 we present the approaches we developed in order to satisfy the different requirements for the identified application scenarios. We state the characteristics that allow satisfying those requirements and give explanations and evaluations.

Chapter 9 presents a concept to expand the topic of collusion secure fingerprints apart from media watermarking. We explain our approach to individualize hash table databases using fingerprinting codes.

In chapter 10 we consider our concepts, for which the requirements of the application scenarios identified in chapter 3 are satisfied by the approaches described afore. For each approach we show how it enables collusion security for a distinct application scenario.

As we could not solve all challenges in the research of fingerprinting codes and collusion security, more research is to be done in different areas. Chapter 11 reflects open challenges and shows corresponding opportunities.

We close this work with a summary of the work elaborated in this thesis in chapter 12.

The appendix covers different test results and proofs as well as further concepts and explanations that confirm the characteristics of our approaches. Details in digital watermarking are not explicated here, therefore we refer to e.g. [100], [31] and [32].

2 Security mechanisms for media

The roots of digital watermarking are entangled with steganography and have a long term history. Steganography was already applied in ancient Greece [56]. In the medieval times watermarking was applied to paper e.g. for letters or paper money. The latter is a famous example for watermarking that is still in use today [105]. Digital watermarking in the current context exists since the 1990s. First approaches that already make use of perceptual models were published in 1996 by Boney *et al.* [24] for audio and Cox *et al.* [28] for audio, images and video respectively.

Collusion secure fingerprinting codes are mathematical codes that can be applied as watermark messages and thereby expand the security provided by digital watermarking. In order to correctly classify fingerprinting codes as watermark messages in comparison to other media security mechanisms, the following sections introduce several terms, definitions and mechanisms established in the media industry.

2.1 Objectives for watermarking and fingerprinting

Information hiding techniques are divided mainly into steganography and watermarking. Whereas steganography is about the secrecy of information transmitted within a cover medium, in watermarking the primary purpose is authenticity or integrity of the cover medium itself by means of the secret information. More general goals also concern access protection, confidentiality, traceability and nonrepudiation.

The most known approaches of digital watermarking deal with copyright protection of digital media. The primal application scenarios were about the copyright protection of still images, shortly afterwards also about audio and video data. At first, these mechanisms were deviated from already in use steganographic techniques, but with the emerge of the digital era watermarking became a field of its own, providing information hiding techniques special tailored for its copyright protection purposes [34]. Note that digital watermarking is a technique that is also applied for purposes other than security. For example, emerging topics are the application of watermarking for *broadcast monitoring* or so called *second screen* applications [78].

In broadcast monitoring watermarking is used e.g. as a counter mechanism to control if an advertisement is aired by TV stations and to count the number of times it is aired. To this effect the watermarks are embedded in the corresponding TV advertisement and can be detected by its advertisers to control what is played and when. A second screen application is provided e.g. by watermarks that are embedded in video or audio signals of the TV program and can be detected by private *second screen* devices such as tablet or smartphone. These devices then connect to the corresponding pages of the Internet to provide more information, advertisement or direct buy opportunities about what is currently playing on TV.

However, these application scenarios are not prone to collusion attacks. Thus, there is no need for collusion secure fingerprinting codes, and therefore these scenarios are no longer considered in this thesis.

In order to accurately value the scenarios described in the coming chapters and correctly classify the proposed fingerprinting solutions, this chapter proceeds determining several mechanisms and definitions used in media security applications.

2.2 Security components for media

The possibility to solidly connecting information with a medium that is given by watermarking techniques, provides a sequence of applications regarding media security. More precisely, the security is based on a secret key principle. The secret key is required for the embedding and detection of information, may it be known to one person or party only, or to several parties, e.g. one for embedding, the other for detecting. In the latter case the security must not rely on the secret key only, the use of further security components is inevitable.

In general, the security components can be classified into *active* mechanisms and *passive* mechanisms. Active mechanisms try to progressively prevent violation of the security object and try to give notice whenever a violation has occurred. Passive mechanisms only serve as control mechanisms, e.g. to ascertain a violation or to ascertain that no violation occurred.

In the following is listed a short description of security components that can be combined with digital watermarking. For a more detailed description see e.g. [25].

- **Encryption:** Encryption mechanisms are divided into symmetric (or secret) key and asymmetric (or public) key mechanisms. Symmetric key algorithms use the same secret key for encryption and decryption. A well known example for block cipher encryption is the *Advanced Encryption Standard (AES)*-algorithm certified by the National Institute of Standards and Technology (NIST). A common drawback of symmetric encryption schemes is that the key has to be somehow transmitted and thereby is prone to attacks. This limits the application to scenarios where transmission is secure and/or not needed at all. Asymmetric encryption schemes use different keys for encryption and decryption and are therefore more secure regarding the transmission of the key. However, this increment in security goes hand in hand with a massive increment of computational effort. This limits its application scenarios to small amount of data to be encrypted. A compromise that is widely applied is given by so called *hybrid* mechanisms, where the advantages of both symmetric and asymmetric encryption schemes are combined.
- **Checksums:** Checksums are mathematical mechanisms to verify if data is original or if it has been changed. Therefore, a short trigger value is somehow computed over a long bit sequence. To detect an error or if the data has changed, e.g. during transmission, the equivalent computation is conducted to receive the same trigger value again. If the resulting value differs, there must be some errors. Regarding watermarking applications, a commonly used checksum is represented by the *Cyclic Redundancy Check (CRC)* [113]. It is based on polynomial division, more precisely a polynomial generator divides one polynomial by another, the remainder term is added to the data. Thereby one can control if the data has changed, e.g. during transmission, by simply computing the same operation again. If the remainder term is the same, the transmission was sound (up to a certain error probability). Obviously, the polynomial generator has to be known to both parties of the transmission.
- **Hashing:** Cryptographic hashing functions are more complex but also more secure than checksums. To prove the integrity of a medium, a hashing function, e.g. *MD5* or *SHA* [97], computes a hash value of fixed length from the medium. The hash value is also known as *fingerprint*, however, to avoid confusion, we prevent from this nomenclature during the further process of this thesis. Important properties hashing functions are subjected to are efficiency, collision-prevention and one-way property. The hash values ought to be efficiently computed no matter what the medium is about nor what size it is. It has to be

very unlikely that two media are allocated to the same hash value. This event is referred to as collision (not to be mistaken by the word *collusion* that plays a major role throughout this thesis). In addition, it has to be extremely unlikely to find the medium by its hash value. Due to its functionality in only one direction, this property is referred to as *one-way function*.

- **Robust hashing:** Robust hashing algorithms are applied in scenarios where cryptographic hashing functions fail to work properly. Cryptographic hashing functions have the property that for example, dealing with multimedia, a cryptographic hash of an image is useless after common image processing operations, as even small imperceptible changes within the image will result in a complete different hash. A robust hash is built to resist common operations on the media file up to a certain degree. The overall goal for robust hashing algorithms is to output the correct hash as long as the perceptual content of the corresponding medium is still distinguishable [38], [5]. However, this high level of robustness leads to an enormous increment of computational cost compared to cryptographic hashes. Application scenarios for robust hashes are among others *Blacklisting* and *Whitelisting* of sensible databases. For example, using robust hashing one is able to quickly scan through digital storage verifying if child-pornographic material is present or not. Using robust hashes suspicious material can be identified no matter of the operations conducted to the multimedia files [108].
- **Error correction codes:** Error correction codes (ECC) are meant to detect, and if possible resolve, errors that can occur during transmission. Therefore, similar to CRC, redundancy bits are added to the data. One distinguishes between automatic repeat request (ARQ) and forward error correction (FEC). The main difference is that ARQ requests the fraudulent data until the message is complete and correct, whereas FEC recognizes and corrects the fraudulent message by means of the redundancy bits. With regards to watermarking, FEC is used to correct errors that arise during transmission or are aroused by media operations and attacks. Two examples for forward error correction are *Convolutional Code* and *Turbo Code* [9]. Most FEC are able to completely correct up to three transmission errors. For common watermark messages this stands for almost 10% of its lengths at cost of only few added redundancy bits.

2.3 Authenticity

The objective for media regarding security is manifold. In general the primary goal is authenticity and integrity [105], yet authenticity is also the objective for fingerprinting application scenarios, and thus for this thesis. Apart from authenticity and integrity there is also access protection, confidentiality, traceability and nonrepudiation defined as security aspects in multimedia [111]. Due to its employment, this thesis restricts itself of defining the aspect of authenticity.

In general, authenticity describes the reliability and originality of a medium. Note that this is not restricted to the digital world, also persons, tools, objects in the digital as well as the analogue world are treat as authentic or not. The so called *authentication* validates if a medium is authentic.

Regarding the media application scenarios of this thesis, we have to distinguish between authorship authentication and customer authentication.

- **Authorship authentication** means that there is some kind of connection to the originator of a medium, e.g. some special feature that clearly identifies its source. Thereby a

medium can be copyright protected, as the originator can be identified. Furthermore, it can emphasize the trust appliers have to the medium, as they trust its source.

- **Customer authentication** is needed to connect a customer with the purchased medium. In case of misuse, the responsible customer can be identified via customer authentication. The neat example for customer authentication regarding this thesis is to individualize the medium according to its customer. This is done via transactional watermarking methods, see the upcoming section 2.4. Thereby, in case of misuse, the individualization feature can be compared to all other individualization features in the originators database, and the responsible customer is uniquely identified.

For both, authorship authentication and customer authentication, digital watermarking can serve as proper features. However, this thesis restricts itself to customer authentication via transaction watermarking methods. The corresponding individualization feature is provided by the collusion secure fingerprinting codes.

2.4 Transactional watermarking

Digital data, such as digital multimedia, are easily manageable and editable. Especially duplicating of media content does not pose any challenge even to inexperienced persons. Besides the obvious advantages for the average user, also malicious users benefit from this. The loss caused by unauthorized redistribution and copyright infringement plays an important role in the digital media industry. This has led to the development of copyright protection mechanisms, such as transactional watermarking.

In general, when speaking about digital watermarking during this thesis, we speak about digital *imperceptible* watermarking. Though, there exists no determinate definition for digital watermarking, the following description generalizes its character:

Using watermarking algorithms one is able to imperceptibly hide secret information in cover-media, as well as to reliably detect and read out this information again.

A typical watermarking scheme can be divided into three stages, the watermark *embedding process*, the *transmission channel*, and the watermark *detection process*. Whether or not the original copy is needed for the detection process one speaks of *non blind* or *blind* watermarking. As the fingerprinting codes proposed in this thesis do not depend on the basis watermarking layer, they are capable of both types. However, as our research group solely deals with blind watermarking schemes, the proposed fingerprinting codes have only been applied onto these in praxis. The following depicts a blind or oblivious transaction watermarking scheme.

Embedding The watermark embedding process requires a copy of the original medium, the watermark message to be embedded and the secret key. The watermark messages applied in Fraunhofer SIT are binary codes. The secret key dictates the positions of the cover media and in which order these are taken for embedding. With this input the watermark embedder generates the watermarked copy.

Transmission The transmission stage concerns the phase in which the watermarked copy has been sold and lies in the hands of the corresponding customer. In this phase the watermarked copy is thus potentially prone to conversions, compressions and attacks. Therefore, the transmission phase is also referred to as *attack channel*.

Detection With the same secret key as used for embedding, the detector is able to read out the watermark message from a watermarked copy. This process is applied in case an unauthorizedly redistributed copy is found.

Note that, no matter of its application, a watermark is part of the media. Technically, a watermark is a slight modification of the cover media. The embedding algorithm alters some parts of the cover in a way that it stays imperceptible for human perception, but the detection algorithm is able to detect the watermark again. The positions to be altered are dictated by the secret key. Without the knowledge of this key it should not be possible to detect the message, thereby referring to *Kerckhoff's* principle of cryptographic security [58].

With regards to the transactional watermarking scenario this means, that individual customer information can be integrated into the media copy, thereby uniquely interrelating the media copy with the corresponding customer. Furthermore, during the further procedure of the media, the information can be detected and read out again in order to identify the copy or customer. The embedded information represents the watermark message or simply the watermark.

Mirrored to the customer authentication as described in the afore section 2.3, this method is applied as a copy protection mechanism. Anticipating one of the scenarios that will be explained in chapter 3, the vendor or distributor that owns the intellectual property right of some media content applies transaction watermarking techniques in order to embed a customer-individual watermark message into every copy sold. In case one of his customers that legitimately purchased such an individually watermarked media copy unauthorizedly redistributes his copy, e.g. on the Internet, and this copy is found, the distributor is able to detect and read out the watermark message again and by this means identifies the responsible customer. As the watermark is part of the media, (to some extend) this also holds after digital to analog conversion and back (DA/AD conversion), for example the process of re-recording *over the air*, e.g. re-recording a movie screened in the cinema via sneaked in camcorder. As this process of screening and re-recording is no longer restricted to a digital data transfer and thus digital control mechanisms cannot effectuate protection, this is called the *analog gap*. Note that this is not restricted to cinema applications. Besides movie re-recording the analog gap also includes music recording, e.g. from radio, photographic shooting of still images, re-writing of written text (whether in digital or physical form), etc. Consequently, this is a distinct challenge for the multimedia industry. Robust watermarking technologies appear to be the only solution capable to resist the analog gap.

In order to do so, the embedded watermark has to survive various processing operations such as common transmission procedures or ordinary conversion operations as well as rigorous attacks that aim to render the watermark useless. This means, each type of watermark is subjected to certain properties which the corresponding watermarking algorithm must provide. The various application scenarios require different settings regarding the particular watermarking properties. *Dittmann* in [32] and *Cox et al.* in [30] agree upon the most important ones. Therefore, these are briefly introduced in the following.

Transparency A watermark message is called transparent or imperceptible, if an average human perception is not able to perceive the difference between the original and the watermarked data. It is not eligible, that the embedded watermark message generates an acoustic or visual perceptible difference to the original for an average hearing or vision. It is called transparent if there can not be recognized any difference between the original and the watermarked copy.

Robustness A watermark message is called robust, if it can be detected reliably even if the data has been modified but not completely destroyed. With these modifications is meant for example image or sound processing, e.g. amplification, mp3, JPEG, and contrast enhancement. It is not meant modifications in terms of attacks, based on the knowledge of the processes of the watermarking algorithm (therefore see item *Security*). The robustness defines the probability of a successful embedding for the one who embeds the watermark.

Capacity To quantify how much information is embedded into the original and how many watermark messages are allowed parallel in the data, is referred to as capacity. According to this, a watermark message should consist of so few bits that it is possible to embed it several times. This is necessary to ensure that the output of the detection process is the proper watermark message, and there is not by chance read out something wrong.

Security In terms of intentional attacks on the watermark, it is called secure, if the embedded watermark message can not be destroyed, detected or falsified, even if the attacker knows the watermarking algorithm and possesses at least one watermarked copy, but does not know the secret key. Note that the distinction between robustness and security changes according to the different application, throughout this thesis the intention of the operation is the crucial part. In case the intention of the operation done to the media targeted to manipulate the watermark per se, this is allocated to security. For example in the video game scenario, a conversion or compression operation is an untypical operation one usually would not conduct. This is seen as an ordinary attack and thus belongs to the security property, whereas the same operations done to e.g. images typically are considered as robustness challenges.

Complexity The complexity determines if the original is needed for the detection process as well as it describes the effort to embed and detect the watermark message. This is mainly dedicated to real-time applications.

Verification This declares, whether only a specific person or group, for example the distributor, can detect the watermark message, because he knows the secret key. Or if the watermark can be detected publicly because the algorithm and the key needed for the detection process is publicly available.

Invertibility One speaks of invertibility, if it is possible to remove the watermark message off the data, to reconstruct the original. The removal can be conducted after the detecting process as well as in between.

Apart from the security property, all listed properties are taken for granted for the further procedure of the thesis. Thus, collusion secure fingerprinting solely is a measure of security. However, chapter 5 also describes the interaction of the watermarking properties transparency, robustness, capacity and complexity with regards to collusion security of fingerprinting codes.

Transactional watermarking is sometimes also referred to as *active fingerprinting* [106], yet to avoid confusion with the word *fingerprinting* this thesis obviates from this definition.

Besides the transactional watermarking method that was described in this section, there exists a variety of different applications for imperceptible watermarking that define the corresponding type of watermarking method. Some are listed in the following:

Integrity watermarking for instance is used in order to verify if the cover content has been tampered or not. That is, in case the sensitive or *fragile* watermark that has been embedded cannot be detected correctly, one assumes the copy has been somehow manipulated and hence its integrity cannot be verified [42], [116].

Copyright watermarking is applied if the copyright holder wants to assure the work to be his intellectual property. Therefore he embeds a distinct watermark message, e.g. a signature, into the media copy. In case his copyright is distrusted, the copyright holder is able to prove his copyright by detecting the embedded watermark again [30], [34].

Broadcast watermarking is not a protection mechanism in the same tenor, but rather a control mechanism. However, the technologies are alike, the watermark is embedded into the content in order to be detected as a clear sign e.g. if the correct content is aired. With it broadcasters and content owners, e.g. TV-stations, have the possibility to track where and when their content is aired and if it is actually aired or not [57], [29].

Second screen watermarking is not a protection mechanism at all. It is a method that allows to provide further information to users. That is, a watermark is embedded into the content, when the content is aired on a *first screen*, e.g. a TV or an image, the user is able to detect the watermark again by means of a *second screen*, i.e. a mobile device having Internet ability such as a smartphone. Having detected the correct watermark, the mobile device manually or automatically links to the corresponding website, that provides the additional information. Thus, second screen watermarking somehow can be seen as imperceptible *QR-codes* [1].

Thus, in concordance to other media processing operations such as compressions or attacks, watermarking also is a procedure that slightly alters the content. Therefore, in order to avoid confusions with the words *modification* and *manipulation*, we determine distinct allocations for them. In the further procedure of this thesis in terms of the watermarking process we speak of a *modification*. A *modified* copy in this manner is thus a copy of the original that has been watermarked. On the contrary, with a *manipulation* is always meant a malicious process on a watermarked copy attempting to attack the watermark. Therefore, a *manipulated* copy always stands for the product of an attack, i.e. a copy that has been tampered in order to prevent the possibility of tracing back.

2.5 Collusion attacks

In general, a transaction watermark has to survive a number of different kinds of media operations and attacks. As the watermark is part of the media, all operations done to the media cover also affect the watermark. Format conversions should not harm the reliability of a robust watermark. Moreover, the watermark should be constructed to survive moderate compression operations. Routine duties for watermarks include compression operations such as moderate mp3 for audio or moderate JPEG for images. If the quality is not significantly degraded by the compression, the watermark must remain detectable.

Nowadays watermarking algorithms are capable to manage these operations and conversions on a satisfactory level. However, the watermark is also prone to attacks that intend to manipulate the watermark itself. A secure watermark ought to resist attacks such as mirroring, rotating, bending and further geometric attacks as well as rerecording via the *analog gap*. For more information regarding attacks on digital media watermarks see among others [27], [46].

Providing an individual message embedded in every copy sold, state of the art transaction watermarking can hardly be circumvented by a single attacker. However, in today's society of clouds and networks, more complex attacks can be mounted. Consequently, groups of professionals have the possibility to apply sophisticated collusion attacks to counter transaction watermarking.

To conduct a collusion attack at least two copies of the same content are needed, where each copy contains individual watermark information. Due to this individual watermark information, the copies, i.e. some parts, slightly differ from each other. Thus, comparing different copies and subtracting their content information exposes watermark information. In accordance with this watermark information, an attacked and manipulated copy could be created providing the same quality as the responsible watermarked copies. In addition, the manipulated watermarked information that the detection algorithm would read out from this manipulated copy is potentially no longer capable to trace back to the responsible copies. Even worse, the manipulated watermark has a distinct probability to trace back to a copy that did not partaken in the collusion.

Simple examples of collusion attacks, that are easily conducted are inter alia averaging the content information or splitting the content and taken parts from every colluding copy. More sophisticated collusion attacks are explained in chapter 4.6. In case the attackers, in this manner also referred to as *colluders*, prevent from applying media processing operations that effect beyond the watermark information, the quality of the resulting media is not degraded. Therefore, the attack is a pure collusion attack. A general assumption that describes these pure collusion attacks is the often cited *marking assumption* introduced by Boneh and Shaw [23]. Following this assumption, for reasons of quality, the colluders only manipulate at exactly the positions that were detected via comparison as described above. The marking assumption and its consequences, as well as sophisticated pure collusion attacks are devoted to chapter 4.

2.6 Security components against collusion attacks

Besides the security components for media described in section 2.2, whether they are applied in conjunction with watermarking techniques or not, they fail to work in a collusion attack scenario as described in section 2.5. Regarding the scope of this thesis, this section is devoted to the security components for media that can be applied in case of collusion attacks apart from collusion secure fingerprinting codes [109].

- **Prewarping:** Prewarping intends to protect from collusion attacks by decreasing the perceptual quality of the resulting collusion attacked medium. For this purpose, each copy of a content to be distributed is individually *warped* using local geometric distortions, e.g. imperceptibly stretching and shrinking parts of an image. These distortions are marginal and therefore do not degrade the quality of the cover media. However, in case two or more customers of the same warped content conduct a collusion attack, the resulting copy is strongly distorted. The distortions are typically strong enough to render the copy useless. One successful example for image prewarping is given by Fenzi *et al.* [36]. By means of *Markov Random Fields* and automated image post-processing the images are prewarped imperceptibly. On the contrary, the unauthorized image copy resulting from a collusion attack is strongly blurred and assumed to be of no merit for unauthorized misuse. In order to increase the protection against collusion attacks, prewarping can be combined with collusion secure fingerprinting codes. This is necessary in scenarios where the prewarping strength is required to be low, whereas the number of attackers are relatively high and the (passive) psychological protection by means of fingerprinting codes is not supposed to suffice.
- **Collusion secure watermarking:** Collusion secure watermarking [15] is a technique for transaction watermarking schemes to resist against collusion attacks at no cost regarding the watermark message length. The difference to common transaction watermarking schemes is its specific watermark message embedding strategy. Two major differences can

be exposed. The first is the use of customer dependent keys for embedding, also referred to as *key switching*. With it, every transaction watermark message is embedded with an individual key, so that the (frequency) coefficients for e.g. *Patchwork* embedding alter from the coefficients used for the messages of the other customers. This can be seen as a security feature. In addition it also increases the robustness of the watermark. The second major difference is that the coefficients for embedding are no longer time- or frequency band dependent. In common watermark embedding schemes, the bits are embedded in sequence to e.g. the time axis. In our collusion secure watermarking approach the information for each bit is spread over the whole content solely depending on the correct individual key [15]. The result is that a colluded copy probably traces back to all colluding customers. This is because for every colluder holds that the correct individual key will output his individual watermark message, or at least parts of it, and thus framing innocent customers is (up to a certain extremely low probability) impossible. The huge disadvantage of this anti collusion technique is its enormous effort for embedding and detection and moreover the liability caused by the huge number of different keys. Even worse is its restriction regarding practical applications. The container embedding strategy by *Steinebach et al.* e.g. [107] that provides fastest embedding, even embedding on the fly that is up to 1000 times faster than playback speed [126], does not work using collusion secure watermarking as published in [15].

2.7 Summary

We gave a short introduction into the objectives that are trailed by watermarking schemes. Especial focus was laid to authenticity that was defined and subdivided into authorship authentication and customer authentication. The latter objective is of major interest for this thesis. Therefore, this chapter briefly introduced digital watermarking in general and its applications as well as common watermarking properties in particular. Hereby, the focus was laid to transactional watermarking due to its prime importance regarding customer authentication. Apart from that, we described in short several security components for media that whether compete or can be applied in conjunction with watermarking. Some are generally important regarding media security, others are additionally required to solve weaknesses in the fingerprinting scenarios due to weaknesses of the underlying watermarking algorithms, e.g. checksums or robust hashing. We will refer to them if adequate protection solely by watermarking and fingerprinting is not possible. Last but not least, this chapter gave a first introduction into collusion attacks and we specified important anti collusion techniques that are used with watermarking. Though these do not provide the assertiveness and potency than collusion secure fingerprinting codes. However, if procurable, some scenarios may give the opportunity to combine these techniques with collusion secure fingerprinting codes.

3 Application scenarios for fingerprinting codes

The initial point for this thesis is given by different scenarios for the application of collusion secure fingerprinting codes. This chapter describes these scenarios in short and explains its relevance in practical applications. Furthermore, the necessity of transactional watermarking solutions and especial the necessity of collusion secure fingerprinting codes are emphasized.

3.1 General classification of the application scenarios for fingerprinting codes

To the best of our knowledge, there does not exist a general classification approach for a reasonable identification of the different application scenarios for fingerprinting codes. Hence, this thesis introduces a comprehensive distinction of the different application scenarios.

- **Promotion:** The promotion and pre-release phase of a product describes the phase between the almost completed creation and development process and the official start of sale, i.e. release. During this period, the product is valued and rated by external persons or parties, hence this phase is crucial for the success of the product in the official sale. A famous example is journalists valuing a new music album.
- **Online shop:** The online shop represents the classical end-customer authentication scenario. Therefore, the product is transmitted to the customer via download. For a reasonable authentication a distinct identification method, e.g. by credit card, is necessary. Neat examples represent online shops for still images or music songs.
- **Digital cinema:** In the digital cinema scenario the target is no longer a clear end-customer authentication but the identification and authentication of local cinemas or movie processing stages. The name giving example thus is a big motion picture group that distributes their new movie to a magnitude of associated cinemas that ought to be authenticated.
- **Video game:** The video game scenario unites the scenarios for *Promotion*, *Online Shop* and in a wider meaning even for *Digital Cinema*. In addition, there is the early phases for game development before the distinct *Promotion and Pre-release* phase. However, it does not include all fields of the whole video game industry, but is restricted to *Boxed Video Games*.
- **Hashtable:** The hashtable scenario differs from the afore mentioned scenarios mostly regarding its cover medium. A hashtable is a list of cryptographic hashes. Each hash is associated to an entry of a large database. Hashtables are used to quickly scan through large sets of sometimes sensible material in order to find matches. An important example is *Blacklisting*, for which the Blacklist of hashtables is used to scan through big data sets of suspicious data searching for e.g. child-pornographic material directly during a crime investigation.

All applications for fingerprinting codes by means of transactional watermarking for the purpose of copyright protection fit into one of these scenarios. However, these are not disjoint, in many cases the boundaries between these scenarios become blurred and several applications may fit into two or more.

On the other side, beside these, there exist further application scenarios for fingerprinting codes. Yet, these do not apply transactional watermarking, some are not even meant as security mechanisms. For reasons of completeness they are listed in the following.

-
- **Medical group testing:** Medical group testing is a field in medical science concerning the identification of a group of virally-infected people within a very large group or population. By means of blood samples the different persons are identified. As people have contact to each other, their samples can get mixed. Since the mixing process can be seen as an equivalent to a collusion attack, group testing makes use of similar tracing strategies as fingerprinting. Examples determining the analogy are among others [59] and [72].
 - **Database watermarking:** Database watermarking is similar to the hashtable scenario, though it is more complex. Because the entries of a typical database are subject to certain rules and properties, robust watermarking of databases is hardly possible. Moreover the necessity of protection against collusion attacks is not present at all, since the underlying watermarking applications are far from being robust and secure on their own. However, chapter 9 concerns about solutions for hashtables, yet provides concepts for a more general database solution as well.
 - **Multiple access channel communications:** Multiple access channel (MAC) communication is situated in the fields of coding theory and information communication. Whenever multiple parties have access to and send messages over a transmission channel, there likewise occur errors. Whether these errors are typical channel distortions or intentional attacks, applying collusion secure fingerprinting codes allows to allocate the message to the correct party. For more information, we refer the reader to [35], [127] and [120] among others.
 - **Frameproof codes and IPP codes:** Frameproof and IPP codes also belong to coding theory, yet they can also be seen as some kind of collusion secure fingerprinting codes. A frameproof code provides protection for an innocent user of being *framed* by malicious users. Stronger conditions are required for IPP (identifying parent property) codes. That is, if a set of collusions outputs a descendant (a codeword), the colluding parties must have a distinct parent (codeword) that identifies the collusions. More detailed information is given in e.g. [119], [90], [49]. We will come back to the identifying parent property in chapter 6.

3.2 Selection criteria for the considered scenarios

The general objectives for transactional watermarking and collusion secure fingerprinting were examined in chapter 2. Hence, this section regards the scenarios that receive deeper inspection throughout the following chapters. Note that only a representative selection of potential application scenarios is considered. The following criteria served as basis for the chosen scenarios.

- **Relevance in praxis:** Due to its intention to support real world scenarios, the practical relevance is of utmost importance. Thus, only scenarios with fingerprinting parameters required by the actual industry are taken into consideration. Hence, all proposed fingerprinting solutions are capable of being applied in practice.
- **Acquisition to watermarking applications:** The whole work is relating to the underlying transactional watermarking applications and algorithms. As all proposed fingerprinting codes are intended to be applied in practical applications, hence only scenarios are considered for which the watermarking algorithm allows embedding fingerprinting codes.
- **Broad covering of the security aspects and requirements:** We intend to cover all potential security aspects and requirements for fingerprinting codes that come along with

the scenarios considered. However, scenarios providing very similar security aspects and requirements are not considered separately.

- **Existing principles:** Research in collusion secure fingerprinting already has a long history. Various different approaches already existed at the beginning of our work. Especially the proposed fingerprinting codes by *Tardos* in [114] and based upon that by *Škorić et al.* in [121] served as basic principles. These approaches also provided accepted notations and definitions. For generalization purposes, they are adopted whenever possible. However, these approaches do not consider practical application scenarios nor the underlying transactional watermarking technology. Thus the scenarios defined in section 3.1 will serve as basis for a broad covering of fingerprinting application scenarios.

The application scenarios as listed in section 3.1 exist in different states of research. Some are widely studied, but some are at its very beginning. Thus, before designing new fingerprinting codes for the different scenarios, the state of the art of fingerprinting codes for the corresponding application scenario at the beginning of this thesis had to be elaborated. The rest of this chapter is devoted to the application scenarios examined for further consideration.

3.3 Promotion and pre-release application scenario

The protection of intellectual property is not restricted to end-customer authentication as described in chapter 2.3. Already during development of the product as well as in the promotional phase where e.g. journalist for music albums or movies, receive pre-versions of the product to value it, it is necessary to apply protection mechanisms to prevent the product to be spread unauthorizedly e.g. over the Internet.

In general, the original content is handed out to a certain number of persons. Copies of the content are individualized by means of transaction watermarks, in order to allow distinction between each handed out copy. In this scenario the number of persons that receive a watermarked copy typically is known in advance. Knowing how many copies with different watermarks are required is a huge advantage for the technical implementation, as it is possible to exhaust the watermarking properties according to the current scenario. We will come back to this in chapter 5.

In the following some examples are listed that fall into the promotional and pre-release scenario.

- **Musical promotion:** In order to promote a new music album or single track, the publishers hand it out to journalists and some exclusive persons before the official start of sale. These journalists and exclusive persons ought to comment and/or write reviews about the new album or track. Thereby they help to advertise and make rumor about it. In this scenario it does not make a difference if the access to the new album or track is allocated via download, stream or physical medium.
- **Audio books:** Promotion for audio books is not yet that common as promotion of music. However, whether in combination with the promotion of the corresponding physical book or not, there is adequate deployment for audio book promotion, and thus for its protection. The scenario is similar to musical promotion as the cover medium is as well.
- **Ebooks:** Ebooks nowadays typically appear in alliance with the corresponding physical book. As it is easier to copy and distribute the digital medium than the physical book, promotion is mainly restricted for the physical book. Thus, protection of the Ebook in the promotional scenario using watermarking methods is hardly applied.

-
- **Movie/TV promotion:** For the same purpose as in musical promotion, transaction watermarks are also applied in movies and TV productions. The only relevant difference is the video cover medium. Comparing playing time, video carries much more information than audio. However, this does not induce more effective payload for watermark embedding. Though, both media types are contained in movies and TV productions, the video medium as well as the audio medium, can be used for watermark embedding. More effort on the characteristics is spent in section 3.5.
 - **Video game pre-release:** The phases just before the official release of a game are especially sensitive against unauthorized distribution. For the publishers it is of immense importance to keep the game a secret and make only little information, e.g. trailers or short game play scenes, public. Their sales policy thus is to increase tense until the official sale start. In addition, this scenario also includes the whole process of the game development. As a magnitude of people and parties have access to at least parts of the game, leakage detection is essential [12]. Moreover, the process of design, implementation and testing sometimes amounts several years. Thus, a reliable environment is strongly desired. The general scenario of video games in the end phase of development and after release is described in section 3.6.

Digital transaction watermarking for promotional and pre-release purposes has been successfully applied for many years for audio and video [33]. Besides the video game medium, which is a very new field of research, in the promotional scenario we work on solid ground regarding the underlying watermarking technology and acceptance in commercial applications. Moreover, the legal perspective for this scenario is quite clear, because the persons that receive transaction watermarked copies typically have a distinct contract they must sign regarding the restrictions upon the corresponding watermarked content. Thus, this scenario can be seen as the most straight.

One famous example of a movie that leaked and by means of transaction watermarking could be traced back to the responsible is *The last Samurai*. Carmine Caridi, a member of the jury of the *Academy of Motion Picture Arts and Sciences* responsible for the *Oscar's* awards, handed his movie copy to a friend who immediately uploaded it on the Internet. Caridi was expelled from the academy and sued for damages. The fine was the maximum statutory damages of 150K \$ [31].

Note that watermark protection holds only if secure transmission can be guaranteed. In case a malicious third person or party successfully attacks the transmission as a *man in the middle* and the medium was transmitted without encryption, the protection is rendered useless. However, the topic of secure transmission methods and techniques is a huge research field of its own and consequently far outside the scope of this thesis.

3.4 Online shop application scenario

In the online shop scenario different types of watermarking mechanisms found their way into praxis. For example, annotation watermarks are applied in order to add annotations to the cover media, e.g. informational meta data such as title, album or artist. Copyright watermarks serve as authorship authentication method in the sense to embed a distinct copyright notice into the medium to be sold. Customer authentication is assured by means of transactional watermarking, for which an individual identifier is embedded into the medium. As collusion security is only necessary in applications that base on protection via individualization of each media copy to be sold, this work will only focus on transactional watermarking.

The online shop represents the classic end-customer authentication scenario. Digital watermarking technology is applied in order to assure copyright of the legitimate owner of a work also after transmission into the system of the publishing company or directly to the customer. Therefore, the watermark message is embedded into the medium to be sold. The majority of applications require a transparent or imperceptible message. Note that the watermarking technology is restricted to digital data, especially to data that allow modification without immediately destroying the cover work. One of the most common examples for the use of watermarking technology via online shop is the distribution of music, for example in (already compressed) mp3 format.

The transactional watermarking technology that serves this scenario requires similar transparency and robustness properties as the promotional scenario. Most transactional watermarking applications serve this scenario, thus the corresponding watermarking technology has been expatiated widely. However, the legal perspective is significantly more complex, because of the non-checkability of the media after transmission over the Internet combined with the likewise huge number of customers, who are typically identified by credit card only. Moreover, the requirements regarding embedding performance are significantly higher in order to be practical.

The following lists important examples addressing the online shop scenario:

- **Music distribution:** Online shops for music distribution are well known and widely spread. The customer selects a song or album. As soon as he confirms his choice, an individual watermarked version is generated and sent to the customer. Thereby the corresponding audio format, e.g. WAVE (*.wav), or already compressed mp3, is negligible.
- **Audio book distribution:** The distribution of audio books in online shops is common as well and equals the music distribution in most instances. For example, potential cover media formats are the same. Technically, watermarking audio books is proportionally harder due to the higher number of sequences containing silence (on average).
- **Image distribution:** Digital images can be used for various purposes. The copyright holder, e.g. the artist or photographer, that distributes his images can whether use copyright watermarks, transaction watermarks or also signatures as passive protection. There exist a wide use of image watermarks for almost all available formats. Though the use of imperceptible watermarks appears more practical, also visible watermarks are relevant for the industry. However, this is mainly restricted to copyright watermarks, e.g. embedding a visible logo or sign into the image. However, this thesis only concerns the transactional application for imperceptible watermarks.
- **Video clip distribution:** With the distribution of video clips in the online shop application scenario is not meant the distribution of video for TV-broadcasting purposes, but the distribution of short video clips, e.g. for trainee purposes. This commercial selling of video clips in many ways equals the music distribution in Online shop, and has increased its importance recently with the new opportunities of mobile Internet. However, in general the challenge for embedding watermark information in videos is comparably harder than embedding watermarks in the same time range of music. Therefore in many applications, the audio track contained in the video clips are more frequently watermarked than the video track itself.
- **Ebook distribution:** The increment of the ebook market goes in analogy to its distribution via online shops. As ebooks form a relatively new kind of media, its sale is solely construed to online distribution. Common formats are *Epub*, *MOBI* and Adobe's *pdf*.

-
- **Streaming media:** Beside the distribution forms listed above, there is also the online shop scenario of streaming media. The vast majority of the amount of streamed media is reserved to streaming video and streaming audio. The corresponding applications are not immediately addressing the online shop application scenario, but are listed here because the model of end-customer authentication and identification also applies for this scenario. The most prominent applications for audio streaming and video streaming obviously are radio broadcasting and TV broadcasting, respectively, but also online video rental shops (Video On Demand VoD). An example for the need to protect the stream e.g. via watermarking and fingerprinting methods represents the live-transference, audio or video, of sports events, such as the soccer world cup or the Olympic Games. There, the content is streamed *live* via radio or setup-boxes and dishonest users directly upload the stream again to the Internet.
 - **Video game distribution:** The online distribution of video games is getting more and more important. Due to the magnitude of its industry and its complex nature regarding different media types a video game contains coupled with a lack of standards, this scenario is devoted to a section of its own, see chapter 3.6.

Note that watermarking technology is a passive protection method, i.e. it does not prevent a-priori misuse, but it helps to detect misuse a-posteriori. Some content providers however, desire an active prevention of copying and/or unauthorized redistribution. Active protection requires an environment that is closed up and secured against access from outside. These environments are referred to as Digital Rights Management (DRM) systems or Enterprise Rights Management (ERM) systems. However, to our honest opinion, a reasonable application of these systems in the scenario of an online shop is not possible. Thus, DRM or ERM systems are no longer considered here.

3.5 Digital cinema application scenario

Copyright violation in digital cinema applications causes the multimedia industry huge losses every year. The big motion picture studios create a new movie, oftentimes a several million dollar production, and as soon as it appears in the cinemas it is leaked and uploaded on the Internet. The market for the newest blockbuster movie is immense. At the same time, the huge number of persons that have access to it (or to a digital copy) is hardly controllable. A reliable copyright protection is strongly desired. Once the copies are shipped to the cinemas, the copyright holders have no longer control over the content. To protect their intellectual property they can choose between various Digital Rights Management (DRM) systems. These are typically intrinsically tied to the devices that also carry the movie layer, thereby complicating unauthorized usage. But these DRM systems are inoperable against the *analog gap*. Once the movie is screened in the cinemas, it can be recorded again, e.g. by some unauthorizedly sneaked in camera.

For the digital cinema scenario, that is the watermark is potentially embedded into both video track and audio track of the movie, this means that the watermarking solution allows to trace back an unauthorizedly re-recorded movie to the corresponding cinema where it was re-recorded. With the knowledge of the setting where the re-recording takes place, the big motion picture groups can enforce the operators of that cinema to increase their controls upfront and during the time the movie is screened. The salient feature for the digital cinema application scenario is that the watermark message is not embedded in order to identify the source of unauthorized action, as it is in the promotional and online shop application scenario, but to

find out the place where this action took place. Into every copy that is shipped to the cinemas is embedded a unique watermark message that clearly identifies the allocated cinema.

Though robust audio watermarking nowadays has a clear chance to resist the analog gap, for robust video watermarking this still is challenging. One must be aware that in countries where the audio track of a movie is likely to get exchanged the protection might be solely restricted to the video track. Watermarking solely the audio track or solely the video track in many cases is unfavorable, a combination of both is to be preferred. Since the challenge of the analog gap is a huge topic of its own, and because this thesis restricts itself to the application of collusion secure fingerprinting codes, we will not go into further detail here but refer the interested reader to e.g. [45] and derivatives.

Note that the phase sensible for copyright violation in a life-cycle of a movie already starts before the movie is shipped to the cinemas. That scenario is devoted to the promotion and pre-release application scenario already discussed in section 3.3.

The model of digital cinema application scenario is not restricted to movies for cinemas, but can be expanded to other services as well. Every distribution scenario in which a copyright holder provides a service to his customers, that themselves provide this service for their own distribution business, can be somehow classified into the same model as the digital cinema application scenario. Another example, for instance, is a model in which the providers of geographical 3D maps want to control their customers' management of their service. That is they want to know e.g. does the number of applications of this service exceed the number that was originally scheduled in their license contract. In case they find unauthorized usage of their service, they want to know the source, i.e. the customer-company, that is responsible for this unauthorized usage. The same model also applies for the licensing of video game engines. The providers of a video game engine do not want video game developers to unauthorizedly utilize their engine. Whether this unauthorized usage is conducted by a licensee or by an opponent party that pirate outstanding components of the engine for their own business of licensing video game engines. A watermarking method to protect from the undesired usage described above promises a remedy for these service providers.

3.6 Boxed video game application scenario

The video game scenario is a relative new scenario for the application of transactional watermarks and fingerprinting codes. The variety of different kinds of video games, its huge market and the complex and very differing composition of various multimedia components consisting in a typical video game make this watermarking scenario a research topic of its own. In fact, during the time of this thesis there is a doctoral research study in progress concerning this topic. In [12] we stated that the distribution of video games can be divided into *games as a service* and *games as a product*.

- **Games as a Service:** Games as a service describes a video game that is continuously in development and purchased e.g. via a standing order. Contrary to the distribution of e.g. audio files, it is not a one time bargain, but an actively interchanging relationship between customers and developers. Typical examples are *Massively Multiplayer Online* (MMO) games, such as *World of Warcraft* (WoW), *Browser Games*, such as *Forge of Empires*, or *Social Games* as provided on *Facebook* for instance.

Due to the the Server/Client concept, to the best of our knowledge, content protection via watermarking techniques is an open challenge for which scientific watermarking has not

been applied yet. Consequently, collusion security is no issue by now and thus, games as a service are no longer considered here. We refer the interested reader to other works of our group, e.g. [12].

- **Games as a Product:** Games as a product describes the video games that are produced at once and, besides smaller expansions or updates now and then, are finalized with the release date. The game is purchased all at once 'in a box' via over the counter or by download. This makes this branch more similar to the classical multimedia. Typical examples for games as a product are boxed game titles for PC and console platforms, e.g. *The Elder Scrolls V: Skyrim* by *Bethesda Softworks*, *Ubisoft's Assassins Creed* series and each of *Rockstar Games' Grand Theft Auto* offsets. But besides there is also a huge list of Mobile Games that can be characterized as games as a product even as they are available for free, e.g. *Rovio's Angry Birds* or *Imangi Studios' Temple Run*. Certainly, the significant technical advances regarding power and performance of mobile devices have blurred the boundaries between boxed games for PC and console and boxed games liable to pay costs for mobile devices. For example, the games *Real Racing 2* by *EA* or *The Walking Dead* by *Telltale Games* were planned for mobile as well as PC or console games. Moreover, there are also huge production games that are exclusively manufactured for specific mobile devices. For example, *Ubisoft's Assassin's Creed III: Liberation* is an exclusive title for *Sony's* mobile platform *Play Station Vita*. This thesis confines itself to games as a product. More precisely, only classical boxed games for PC and console are considered for further investigation. This is because the classical low budget productions for mobile games are available for free, and consequently there is no market for unauthorized redistribution and thus for copyright protection including collusion security.

The development of a typical boxed video game is a process of sometimes several years. Within this time a magnitude of in-house capacities as well as professional external testers have access to the game. Additionally, specialized focus groups are hired for the testing process closely before the release. All testing participants and everyone that has access have signed non disclosure agreements, hence unauthorized behavior equals committing a crime and is legally prosecuted. The watermarking scenario considering unauthorized behavior before the official release is devoted to the promotional and pre-release scenario described in section 3.3 and elaborated in chapter 5.1.

To the best of our knowledge, the scenario of watermarking video games was primary aroused within our research group, see [12], [118], [13] and [70], in parallel to the work on collusion secure codes for this thesis. There is no state of the art nor practical experiences for the end-customer authentication via watermarking techniques. On first sight the watermarking deployment appears as obvious. As a video game inter alia consists of different components from classical multimedia, e.g. audio files and videos, in combination with rather new components (regarding watermarking) such as 3D models and texture images, deploying watermarks in each component is possible, see figure 1. Moreover, the magnitude of files of each component, e.g. ca. 19000 texture images and about 3000 3D models in the video game *The Elder Scrolls: Skyrim*, promises huge payload for embedding. This rises new opportunities regarding the watermarking properties but also induces new technical challenges. One challenge is the watermarking of new cover media formats, such as *Microsoft's* image compression format *Direct Draw Surface* (DDS) which is applied in almost every game. Another challenge is the handling of watermarking not only for single files as it is common in other types of multimedia, but for rather huge sets of the same cover media sometimes containing several thousands files. As the research question that is addressed for this thesis is restricted and exhausted with collusion secure fingerprinting

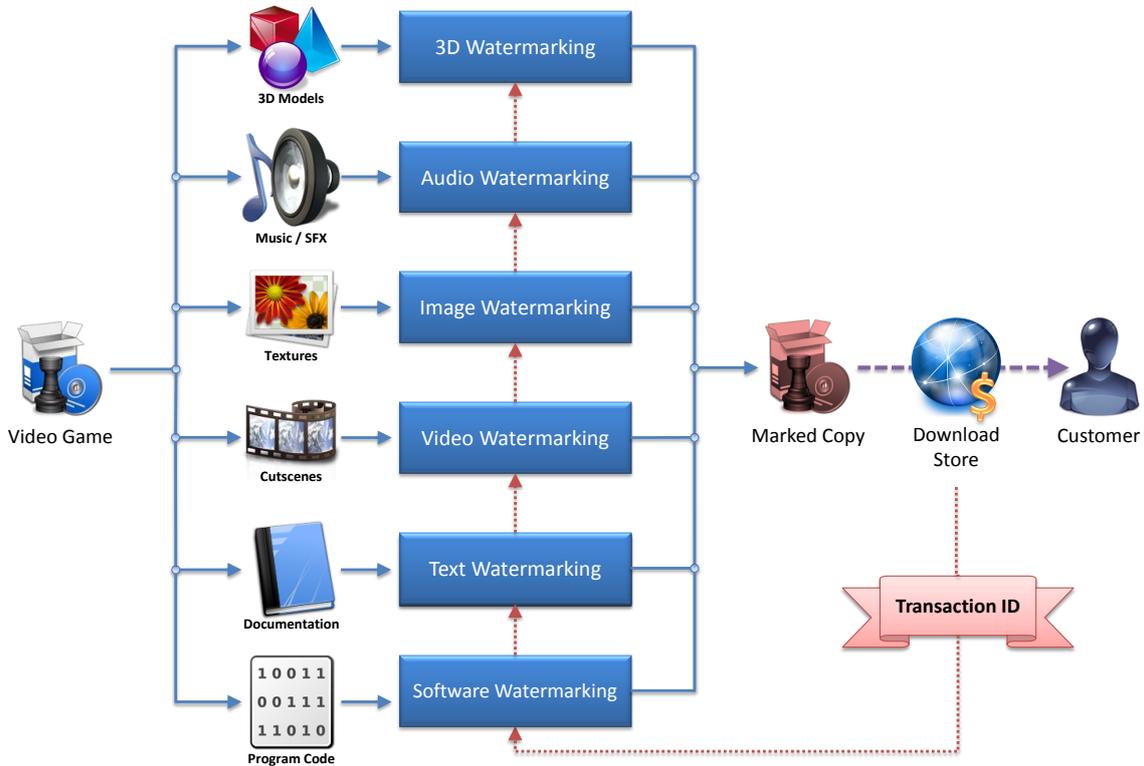


Figure 1: A boxed video game consists of different types of media components that can be watermarked, based on [117]

codes and its application scenarios, the technical basis regarding video game watermarking is assumed as solid. In this thesis the focus is on the deployment of fingerprinting codes for video games. For the technical challenges regarding video game watermarking, we refer the interested reader for instance to the work of [12] and to a dissertation that is elaborated in parallel to this thesis (but not published yet).

Copyright protection is as common as unpopular in the video game community. As the value of money of a typical boxed game is significantly higher than of other media goods, the market of free illegally purchasable versions is immense. The effort and costs in creating strong DRM protection mechanisms is huge. However, the so called *release groups*, i.e. organized groups of hackers that compete against each other in order to be the first to crack the protection, spend huge effort as well. In most cases, the result is that the protection is broken shortly after release of the new game. Thus, end-customer authentication realized by watermarking methods is strongly suggested. With it, cracked versions of a game could be traced back to the responsible persons or parties, i.e. to the release groups. As soon as the cover of one release group is blown, the others might refrain from unauthorizedly redistributing watermarked content. Therefore, transactional watermarking technology ought to be applied in all video game components possible. Furthermore, the watermark needs to be spread over the whole content in order to exacerbate braking the protection.

3.7 Database/hashtable application scenario

The leakage of databases is an important issue wherever databases are exchanged. With the possibility to detect the leakage, the illegal distribution could be constricted directly at its root before the content arrives at the hands of numerous end-users. Embedding fingerprinting codes provides a method to individualize databases and thereby allows to detect a leak. There exists a variety of different databases. However, this work is reduced to datasets of cryptographic hash databases.

Collections of cryptographic hashes appear in several application scenarios. In IT-forensics, cryptographic hashes allow to efficiently scan through big data sets in terms of *blacklisting* or *whitelisting*. Blacklisting describes browsing through big data sets searching for illegal content. Whitelisting describes the process of filtering legal content out of big data sets.

One example for blacklisting is the generation of hash tables to scan quickly for child-pornographic material during forensic examinations for a first evaluation of evidence. Large whitelists for example provide cryptographic hashes of all components from some known software.

A useful application scenario could be a company, whose in-house firewall automatically prohibits access to illegal image files with the help of such a list. Furthermore, anti-virus programs could benefit from these hashes by blocking or deleting illegal data material.

Besides these advantages of cryptographic hashes, there also exists a high risk that these lists are misused. A person who has access to a blacklist could easily determine whether specific data is included in this list by computing the hash of the data and using it to scan through the list. This makes cryptographic hashes a strong tool for misuse, e.g. to efficiently search for child-pornographic material or for weapon and bomb construction sets. In fact, many Internet-crawler provide searching for content whose hashes match to the hashes of a blacklist and thereby potentially access its content. For this reason blacklists are rarely distributed.

Solutions for the hashtable scenario applying information hiding techniques are rare. Though there exist some approaches in order to watermark databases or approaches that add a *salt* to the hashes.

A first well-known database watermarking scheme for watermarking numerical values in relational databases issued by *Agrawal* and *Kiernan* [3] is based on the assumption that the watermarked database can tolerate a small amount of errors. For watermark embedding, a least significant bit approach is used, where numerical values are changed. The work in [104] is an extension for categorical data. Here a categorical attribute is selected and changed to other values of the attribute e.g. *blue* is changed to *yellow* if this change is tolerable in certain applications. In [68] the authors introduce a distortion-free scheme for watermarking categorical data, where database depositions are arranged in a key-dependent way. Their results are high probability detection and also the possibility for localization of modification in the watermarked data. However, for cryptographic hashes this approach is not possible. If a cryptographic hash is modified, it is useless for white- and blacklists. Distortion-free approaches are easy to obviate by e.g. a random hash permutation. After the permutation the watermark detector will not find the embedded information while the database is still the same.

Salting is a well-known alternative to individualize cryptographic hashes. A random binary sequence is added to the data to be hashed or encrypted to ensure that the hash or cipher message differs from other versions made from the same data. When used with hashing, the salt

is made public to enable the verification of the hash. The primary goal of salting is to increase the security of passwords by making lookup-tables impractical to use [98]. In combination with cryptographic hashes salting also can be used to individualize those hashes by simply adding the salt to the data to be hashed [17]. Thereby common files like OS libraries cannot be recognized in salted hash sets without re-calculating their hash together with the known salt. This makes the process of recognition much more complex. Salting, on the other hand, also increases the complexity of the hash-set generation. In case of massive data collections re-hashing all data with a salt would be very expensive. Even just adding the salt to the original hash and re-hash both can be a time-consuming operation when many copies of a hash set consisting of 100,000 files or more are distributed.

For this reason, a fingerprinting approach is desirable. Individualizing each hashtable by applying collusion secure fingerprinting codes promise a security level that finally allow distribution of blacklists or whitelists. Requirements needed for the application of for example *Tardos* codes [114] or its derivatives, e.g. [121, 86] are discussed in chapter 5.5.

4 Collusion secure fingerprinting codes: Concepts and properties

The weaknesses of random generated transaction watermark messages against collusion attacks combined with the effortlessness to create such attacks lead to the development of sophisticated codes that, applying these as transaction watermark messages, promise resistance against collusion attacks. Due to this property and because transaction watermarks in its early days were also denoted as *active fingerprints*, e.g. [23], these codes were named *collusion secure fingerprinting codes*. Not to be confused with *passive fingerprinting* alias robust hashing as briefly described in chapter 2.2.

However, the history of these codes began even before digital watermarking became popular. Already in the 1980s mathematical codes were published that provide minor collusion resistance, i.e. that are collusion resistant to some extent, [125], [20]. Already at that time these codes were referred to as *fingerprinting*, as their main purpose was identification of different users.

This chapter introduces the general definitions, parameters and properties that are needed dealing with collusion secure fingerprinting codes in detail. Section 4.1 concerns the general fingerprinting scheme, in section 4.2 we list the different types of fingerprinting schemes and state some examples. The most important parameters and definitions are explained in section 4.3 in accordance to the parameter description in state of the art literature. Section 4.4 gives the definition and explanation of the *marking assumption*, an assumption most fingerprinting codes in literature as well as our approaches rely on. This is followed by a description of the so called *Tardos* scheme by Tardos [114] and accordingly the *symmetric Tardos scheme* by Škorić *et al.* [121]. Afterwards we list some sophisticated collusion attacks, i.e. independent of the watermarking application, in section 4.6. Section 4.7 closes this chapter with a short summary.

4.1 The general fingerprinting scheme

A fingerprinting scheme in this manner represents the whole system that concerns the protection of a digital product. Therefore it can be divided into five independent processes, as enumerated in the following:

1. **Fingerprint Generation Process:** The first process of the scheme is the generation of the fingerprints. A fingerprint is a codeword that is generated to serve as a clear identifier for a user or source. The fingerprints need to be generated in advance. That is because, contrary to randomly generated transaction watermark messages as described in chapter 2.4, most codes are generated under certain conditions and probabilities. For example, the number of fingerprints to be generated and their lengths must be known. Hence, the result of the fingerprint generation process is a codebook of a set of codewords, a list of fingerprints respectively.
2. **Fingerprint Embedding Process:** As soon as a customer confirms purchasing a media copy of content, the fingerprint embedding process starts. Therefore, one of the generated fingerprints is assigned to the customer. This fingerprint is transformed into its watermark information and embedded into the current media copy in analogy to the watermark embedding process as described in chapter 2.4.
3. **Collusion Channel:** The collusion channel represents the phase in which a fingerprinted media copy is not in the hands of the distributor or copyright holder. Thus, it starts as soon as the fingerprinted media copy is sold to a customer. By that time the distributor has no

longer control of this copy. In the collusion channel the fingerprinted copy is potentially prone to manipulations and attacks. In case of no unauthorized action, no unauthorized copy can be found and thus, the psychological protection by means of discouraging from unauthorized actions via fingerprinting was successful and the fourth and fifth process of the scheme need not be called at all. On the other hand, in case an unauthorized version of the media content is found the distributor or copyright holder initiates the next processes.

4. **Fingerprint Detection Process:** The fingerprint detection process tries to detect and read out the fingerprint contained in this copy. More precisely, a watermark detection process as described in chapter 2.4 detects the watermark information, transforms this information into the corresponding fingerprint code and compares it to all fingerprints that have been assigned. If the fingerprint is equal to one of the fingerprints that have been assigned, the responsible customer is found and the scheme is completed. On the contrary, if there is no unique match, it is assumed that the copy underwent a collusion attack. In that case, the final process is called.
5. **Fingerprint Tracing Algorithm:** The last process of a fingerprinting scheme represents the tracing algorithm. The tracing algorithm receives the fingerprint that was detected from the unauthorized copy, the list of all fingerprints that have been embedded into distributed copies, as well as all the other information from the fingerprint generation process, e.g. fingerprinting parameters and embedding distribution as defined in section 4.3, etc. By means of this information it is evaluated which fingerprint(s) of the list is responsible for the collusion.

Note that, the processes two through four, i.e. the fingerprint embedding process, the collusion channel, and the fingerprint detection process, represent a classical watermarking scheme. Therefore, the technique behind the embedding process as well as the detection process is not topic of further consideration. As already mentioned in the introduction chapter, this work assumes total sovereignty of the basis watermarking scheme, in order to focus on the challenges that are specific for fingerprinting exclusively. Moreover, the transformation of fingerprints to watermark information and its inverse transformation is negligible for this work. That is because the majority of watermarking algorithms, including all algorithms of Fraunhofer SIT, operate with binary watermark messages, and for that reason all fingerprinting codes proposed in this work are binary codes as well. In addition, the collusion channel is equal to the attack channel as described in chapter 2.4. Thus, the first and fifth process, the fingerprint generation as well as the tracing algorithm respectively, are fingerprinting specific. Consequently, the fingerprinting solutions proposed in chapters 6 through 9 typically consider both, a fingerprint generation process and a suitable tracing algorithm.

4.2 Types of fingerprinting schemes

Fingerprinting schemes are classified as whether probabilistic or deterministic, and whether static or dynamic. The following definitions explain the corresponding context.

Definition 1 (Deterministic Fingerprinting Scheme) *A fingerprinting scheme is called deterministic, if it is able to prove with certainty that its output fingerprint actually is a colluder-fingerprint.*

With *colluder-fingerprint* is meant a fingerprint that is assigned to a colluder that has partaken in the current attack.

Definition 2 (Probabilistic Fingerprinting Scheme) *A fingerprinting scheme is called probabilistic, if it is able to prove within a certain error probability that its output fingerprint actually is a colluder-fingerprint.*

Note that these definitions do not consider the case for which the fingerprinting scheme does not output any suspicious fingerprint at all.

Regarding the significance in practical applications, especially regarding legal issues, deterministic fingerprinting schemes provide a stronger statement compared to probabilistic schemes that always provide a probability of failure. However, probabilistic fingerprinting schemes are the most appropriate for applications in practice, since the error probability that is permitted in general allows to generate significant shorter fingerprints. Therefore, all fingerprinting schemes but one proposed in this thesis are probabilistic. In chapter 6.1 we will also propose one outstanding deterministic fingerprinting scheme which is at least competitive to state of the art probabilistic schemes of the same kind and that is approved capable of being applied in practice [94].

Definition 3 (Static Fingerprinting Scheme) *A fingerprinting scheme is called static, if its processes are called solely once.*

Static schemes are the most common. Each of the processes of a typical fingerprinting scheme, i.e. fingerprint generation, fingerprint embedding, collusion channel, fingerprint detection, tracing algorithm, is called once at most. Additionally, the order in which the processes are called is distinct, from process 1 to process 5.

Definition 4 (Dynamic Fingerprinting Scheme) *A fingerprinting scheme is called dynamic, if its processes are called several times in circles.*

In a dynamic fingerprinting scheme the first or second process can be run again after the end of the fourth or fifth process. In other words, after receiving a first feedback from the fingerprint detection process or tracing algorithm, the fingerprinting scheme uses this information for a second round and utilizes this feedback information for generating adjusted fingerprints and embeds these into the media. In consequence, after the second round, the tracing algorithm is able to output a potentially more reliable statement about which fingerprint is a colluder fingerprint. Again this feedback is taken for a further round and so on. In that way is proceeded until the statement is sufficiently reliable.

Note that applications scenarios for dynamic fingerprinting schemes in practice are very limited. Streaming media provides the only scenario in which utilizing received feedback for further generation and embedding is possible. Furthermore, static fingerprinting schemes are investigated much more and can be transformed into dynamic ones, for instance see e.g [63]. Therefore, dynamic fingerprinting schemes are no longer topic of this work.

In summary, there exist four combinations of the properties defined above. Consequently there are four different kinds of fingerprinting schemes.

We refer the ambitious reader to *Laarhoven* in [66] for a deeper discussion about the different kinds of fingerprinting schemes. Here we only list some examples of the combinations possible.

static-deterministic fingerprinting schemes A fingerprinting scheme is referred to as static-deterministic, if the fingerprint generation process as well as the tracing algorithm is called only once and the output of the tracing algorithm provably identifies one or more colluders. Corresponding examples are frameproof codes, e.g. [112], [19], and codes providing the identifiable parent property (IPP-codes), i.e. codes that find the correct source (*parents*) of e.g. molecules or DNA sequences, [48], [6].

static-probabilistic fingerprinting schemes A fingerprinting scheme is referred to as static-probabilistic, if the fingerprint generation process as well as the tracing algorithm is called only once and the output of the tracing algorithm with a high probability identifies one or more colluders. The most prominent examples for this kind of codes are [23] by Boneh and Shaw as well as [114] by Tardos.

dynamic-deterministic fingerprinting schemes A fingerprinting scheme is referred to as dynamic-deterministic, if the fingerprint generation process as well as the tracing algorithm can be called in rounds several times and the output of the (final) tracing algorithm provably identifies one or more colluders. Prominent examples are inter alia Fiat and Tassa in [37] and Berkman *et al.* [16].

dynamic-probabilistic fingerprinting schemes A fingerprinting scheme is referred to as dynamic-probabilistic, if the fingerprint generation process as well as the tracing algorithm can be called in rounds several times and the output of the (final) tracing algorithm with a high probability identifies one or more colluders. The basis scheme was given by Tassa in [115], recently Laarhoven *et al.* published a sophisticated version in [63].

The combination that is explored the most in recent research is the class of static-probabilistic fingerprinting schemes. In the following, if not explicitly stated contrary, the term 'fingerprinting scheme' always represents a static-probabilistic fingerprinting scheme.

4.3 Fingerprinting parameters

As already mentioned in the beginning of this chapter, the fingerprint generation process as well as the tracing algorithm are the processes that are special for fingerprinting within a fingerprinting scheme.

The general understanding of the probabilistic fingerprint generation and subsequently of the different kinds of tracing algorithms are described in this section. At first the definitions and conventions that are required are posted below.

Definition 5 (Fingerprint) *A fingerprint in the context of this thesis always is a unique message code with the purpose to clearly identify a person or data file. Related to coding theory, a fingerprint is a codeword belonging to a fingerprinting code.*

In the procedure of this thesis we restrict ourselves on binary fingerprints, i.e. binary codes to be embedded as watermark messages.

Definition 6 (Fingerprinting Code) *A fingerprinting code is a code book containing a set of coherent generated fingerprints, that provide the collusion resistance property.*

Note that a fingerprinting code that stands on its own is solely able to provide the collusion resistance property. In order to additionally provide collusion security, a fingerprinting code needs to belong to a fingerprinting scheme supplying a tracing algorithm that makes use of the specific information from the corresponding fingerprinting code. Thus, a collusion secure fingerprinting code can be seen as one part of a two-tuple containing the fingerprinting code and its corresponding tracing algorithm.

In accordance to *Tardos* in [114], now and in the following the number of fingerprints, number of users respectively, is denoted as n . The number of positions of each fingerprint, i.e. the code length, is always referred to as m .

Definition 7 (Fingerprinting Matrix \mathbf{X}) *The fingerprinting matrix \mathbf{X} consist of n rows. Each of these rows represents a fingerprint of length m associated to the same fingerprinting code.*

The dimensions n and m of \mathbf{X} , present the number of fingerprints and its length, respectively. Note that to embed fingerprints of code length m , the media cover ought to provide sufficient payload, denoted as M . That is, the amount of watermark information that is needed to successfully embed a fingerprint of length m must be smaller than the available payload M .

In a probabilistic scheme the fingerprint matrix \mathbf{X} is generated according to a distinct distribution that follows whether a continuous density, e.g. [114], [121] and recently [86], or a discrete distribution, e.g. [84], [65].

More precisely, the matrix is filled column-wise and column independent. Thus, the symbols α from an alphabet Λ that are sorted into a column i where $i = 1, \dots, m$ are prescribed by a probability $p_i(\alpha)$. This probability $p_i(\alpha)$ follows a probability density function $f(p_i)$, describing the distribution of the probabilities, i.e. the number of occurrences of the different probabilities p_i for the columns of the fingerprint matrix \mathbf{X} .

Downscaled to the binary case, i.e. $\Lambda = \{0, 1\}$, this ends up to $\alpha \in \{0, 1\}$ and for simplification purposes, we restrict ourselves to a probability p_i representing the probability for a 1 in column i . Thus, the probability for a 0 is represented by the counter-probability $1 - p_i$. In addition, we will often make use of the notation p_{y_i} . This represents the probability of the actual symbol (1 or 0) in position i of a fingerprint y .

The probabilities are generated according to a certain distribution function $\mathbf{f}: [0, 1] \mapsto \infty$ that is either continuous and follows a symmetric curve, or discrete meaning that it has several supporting points. Chapter 4.5 below gives a more detailed description of a continuous distribution by the example of the famous *Tardos distribution*.

The maximum number of colluders the fingerprinting code aims to provide resistance against is another important parameter for the fingerprint generation process. This parameter is denoted as c_0 . For the work with fingerprinting codes it is inevitable to understand, that this parameter must be chosen in advance, already before the code is actually affected by a collusion. Thus, the distributor of media content, who makes use of the protection of fingerprinting codes, has to estimate the number of pirates that potentially cooperate in a collusion. Therefore, he has to choose the maximum number the applied fingerprinting code should be resistant against. Throughout this thesis it is referred to as c_0 .

This parameter has a massive effect on the fingerprinting code and its applicability. It is responsible that a distinction of different application scenarios is necessary in order to apply

fingerprinting codes in practice. That is because c_0 massively effects the code length, i.e. the length of the fingerprints. The fingerprint has to fit the given watermarking payload to be embedded by the watermarking embedding process, otherwise the fingerprinting scheme does not provide the desired protection. In most recent probabilistic fingerprinting schemes, c_0 effects the code length to the square. *Tardos* already prove this to be optimal [114].

For simplification purposes, the set of colluder-fingerprints, i.e. the set of fingerprints that have partaken in a collusion attack, in the following is abbreviated with C . The matrix that consists of all colluder-fingerprints is accordingly denoted as X_C . The number of fingerprints within C , the number of colluders respectively, is referred to as c . Hence, X_C is a $(c \times m)$ subset of \mathbf{X} . Note that for a fingerprinting scheme to be sound, it must always hold $c \leq c_0$, the actual number of colluder must be at most as high a the maximum number the code is construed to provide resistance against.

As mentioned before, a probabilistic fingerprinting scheme allows certain probabilities that the algorithm might fail. There are two corresponding errors, the *false positive* error and the *false negative* error.

Definition 8 (False positive error) *In a fingerprinting scheme, the event that the tracing algorithm erroneously outputs a fingerprint that was not part of the collusion is called false positive error.*

In other words, a false positive error occurs, if there appears one or more fingerprints of innocent users within the set of fingerprints that the tracing algorithm identifies as colluder.

Definition 9 (False negative error) *In a fingerprinting scheme, the event that the output set of the tracing algorithm does not contain any colluder-fingerprint is called false negative error.*

In other words, a false negative error occurs, if the tracing algorithm is not able to identify any of the colluders at all. The above definition of the false negative error is taken in the further proceeding of this thesis. It must not be mistaken with the deviating definition saying a false negative error is the event in which not all colluder-fingerprints, but only a subset can be identified.

For the generation of a probabilistic fingerprint matrix, the distributor has to choose in advance an upper bound of the error probabilities that he allows for his scheme. The corresponding rates are denoted as ε_1 and ε_2 .

Definition 10 (The error bound ε_1) *In a fingerprinting scheme, the upper bound for the probability of identifying a specific innocent-fingerprint is denoted as ε_1 .*

This must not be mistaken with η_1 , that describes the upper bound for the probability η that there are innocent-fingerprints among the accused. More explanation regarding η is provided in the appendix A.

Definition 11 (The error bound ε_2) *In a fingerprinting scheme, the upper bound for the probability of not accusing any colluder-fingerprint at all is denoted with ε_2 .*

Note that in most fingerprinting schemes only the ε_1 rate is required for the fingerprint matrix generation process, e.g. [114], [121], [102].

The previously mentioned definitions are related to the fingerprint generation process. The definitions that follow are associated with the fingerprinting tracing algorithm.

The fingerprinting tracing algorithm outputs one or more suspicious fingerprints. This is a subset of the fingerprints from the fingerprint matrix \mathbf{X} or it is the corresponding suspicious users. The output is denoted with Σ . The decision of which fingerprints are suspicious is made according to certain scores between the unauthorized and potentially manipulated fingerprint y and one or more of the fingerprints in matrix \mathbf{X} .

These scores now and in the following will be referred to as *accusation scores* or simply *scores*.

They can be calculated between y and $X_j := \mathbf{X}_{j;1,\dots,m}$, for a single user j , or between y and X_{j_1}, \dots, X_{j_l} for several users j_1, \dots, j_l . Consequently, they are divided into *single accusation scores* or *joint accusation scores*.

Definition 12 (Single accusation score) For every position y_i of the detected and potentially coluded fingerprint y of the unauthorized media copy, a certain position-dependent score is calculated between y_i and the corresponding position i of each user's fingerprint $X_j(i)$, $j \in \{1, \dots, n\}$. For an arbitrary user j , the accusation score S_j is calculated as a certain composition of all position-scores.

$$S_j = \bigcirc_{i=1}^m \varphi(y_i; X_j(i)) \quad (1)$$

Definition 13 (Joint accusation score) For every position y_i of the detected and potentially coluded fingerprint y of the unauthorized media copy, a certain position-dependent score is calculated between y_i and the corresponding position i of a group of user's fingerprints $X_{joint} = X_{j_1}(i), \dots, X_{j_l}(i)$ with users $\{j_1, \dots, j_l\} \in \{1, \dots, n\}$. The accusation score S_{joint} is calculated as a certain composition of all position-scores.

$$S_{joint} = \bigcirc_{i=1}^m \varphi(y_i; X_{joint}(i)) \quad (2)$$

The weight function φ is defined as

$$\varphi : [y_i; X_{j_1}(i), \dots, X_{j_l}(i)] \mapsto \varphi(y_i; X_{joint}(i)) \in \mathbb{R}.$$

Note that for a single accusation score it holds $l = 1$. The composition symbol \bigcirc in definitions 12 and 13 represents the mathematical operation that is applied. In the fingerprinting scheme proposed by Tardos in [114] and in most of its derivations, the symbol \bigcirc stands for the summation operation, thus, the accusation score for each (single) user is calculated as the sum over all position scores of the corresponding user.

Because the tracing algorithms applying a single accusation score calculate the scores for each single user separately, they are called *single tracing algorithms*.

Definition 14 (Single tracing algorithm) A fingerprinting tracing algorithm T that calculates the single accusation score for each user j , $j = 1, \dots, n$ and with it decides which fingerprints are output as potential colluder-fingerprints, is referred to as single tracing.

Most single tracing algorithms use a threshold in order to separate potential colluder-fingerprints from innocent fingerprints, see section 4.5.

Definition 15 (Joint tracing algorithm) A fingerprinting tracing algorithm T that calculates joint accusation scores between y and several users' fingerprints X_{j_1}, \dots, X_{j_l} in correlation, and with it decides which fingerprints are output as colluder-fingerprints, is referred to as joint tracing.

A typical example of generating a joint accusation score is the use of the *maximum likelihood estimator*, e.g. in [74].

4.4 Marking assumption

The so called marking assumption, or marking condition [114], was introduced by Boneh and Shaw in [23]. It claims that the only chance for attackers to generate a high quality copy that cannot be traced back, is to manipulate the copy only in the positions where the watermark is placed. Obviously the attackers have no knowledge about the watermark positions. Thus, in order to manipulate watermark positions the whole media file is affected by common attacks, e.g. compression. Consequently, due to the sophisticated watermarking algorithms and perceptual models applied in the course of embedding, the manipulations that significantly alter the watermark also extensively degrade the quality of the copy. However, as already described in chapter 2.5, the attackers have a chance to detect positions of the watermark when they conduct a collusion attack. Since the watermark messages embedded into each customer's copy are individual, at least some watermark positions differ from each other. These positions can be detected by simply comparing two or more individually marked copies of the same content, by e.g. subtracting their energy or creating the difference-image. These positions carry parts of the watermark information and are thus called *detectable*.

Definition 16 (Detectable positions) Comparing two or more individually watermarked copies of the same content, the positions where the copies differ from each other are called *detectable positions*.

By means of this, the following definition allows a more handy interpretation of Boneh and Shaw's marking assumption.

Definition 17 (Marking assumption) In order to avoid the possibility of being traced back for their forged high quality copy, the attackers only manipulate at detectable positions.

Note that the attackers are only able to detect differences within the corresponding media file, e.g. different positions such as pixel values in images. Neither have the attackers knowledge about which detected difference position belongs to which watermark message bit, nor does one single detected difference position thoroughly represent a watermark message bit. This is because a watermark message bit is represented by a group of modified content values, e.g. a

group of pixel values. One single detected difference position thus is only a part of these content values that represent the watermark bit.

Mapping this to the fingerprinting scenario, for simplification purposes the assumption is made that the attackers are able to detect not only content values but ordinary watermark or fingerprint positions. Also for simplicity we assume that they even know the corresponding symbol, i.e. whether it is '1' or '0', since already *Furon* and *Perez-Freire* proved that there is no distinction regarding the strength of their attack, whether the attackers know the symbol or not [41].

Based on this, the following extension of definition 16 is utilized in the further procedure of this thesis.

Corollary 1 *Be $C = \{X_{j_1}, \dots, X_{j_c}\} \subset \{X_1, \dots, X_n\}$ a collusion of size c . A fingerprint position i is called detectable if holds*

$$\exists X_{j_k}, X_{j_l} \in C, k \neq l : X_{j_k}(i) \neq X_{j_l}(i) \quad (3)$$

A fingerprint position is called undetectable if holds $X_{j_1}(i) = X_{j_2}(i) \dots = X_{j_c}(i)$.

Note that regarding the watermarking application, a small addition to the definition of the marking assumption is necessary. To avoid creating perceptual artifacts within the forged copy, the manipulation of the detectable positions must be restricted to the difference range of the corresponding positions of the individually watermarked copies that are compared. However, for simplification purposes the fingerprinting scenario of this thesis is restricted to the message symbols of the watermark or fingerprint. As this is a binary decision, the watermark detection outputs a binary message respectively, a manipulated position can only result in a flipped binary message symbol. For other detection models see for instance *Škorić et al.* [122], *Boesten* and *Škorić* [22] or *Oosterwijk et al.* [86].

4.5 Symmetric Tardos scheme

This subsection explains probabilistic fingerprinting schemes with the example of the symmetric *Tardos* scheme, consisting of the fingerprint generation process with a continuous distribution function according to *Tardos* in [114] as well as an alternative fingerprint generation with a discrete distribution function. This is followed by the fingerprint tracing algorithm according to [121].

4.5.1 Tardos fingerprint generation

For the fingerprint generation process the distributor needs to choose the parameters ϵ_1 and c_0 to calculate the minimum code length m the fingerprinting scheme mathematically proves to satisfy the chosen error probabilities. The code length formula is given as

$$m \geq d_m c_0^2 \ln \frac{1}{\epsilon_1} \quad (4)$$

The code length parameter d_m depends on the chosen fingerprinting scheme. That is, for the original *Tardos* code [114] d_m is set as 100. Recently it was shown, that in the asymptotic case

the parameter can be lowered to $d_m \geq \frac{\pi^2}{2}$, [64]. However, this is provable secure only for very large and therefore not realistic collusions, that are not considered within this thesis.

With it, the distributor generates an $n \times m$ matrix \mathbf{X} , where the n denotes the number of users to be accommodated in his system. Ergo the j th row of the matrix corresponds to the fingerprint which is later embedded in the copy that is released to customer $j \in \{1, \dots, n\}$. The entries X_{ji} of matrix X are generated in two steps:

First, the distributor picks m independent random numbers $\{p_i\}_{i=1}^m$ over the interval $p_i \in [t, 1-t]$, with a so called *cutoff* parameter $t < \frac{1}{4}$. Each $p_i = \sin^2(r_i)$ is selected by picking uniformly at random the value $r_i \in [\bar{t}, \frac{\pi}{2} - \bar{t}]$ with $0 < \bar{t} < \frac{\pi}{4}$, where $\sin^2(\bar{t}) = t$. The choice for the cutoff parameter t in the original Tardos code [114] was $t = (300c_0)^{-1}$, without giving any rationale. In [121] Škorić *et al.* prove that this was suboptimal, the asymptotical optimal choice is $t \leftarrow \frac{\gamma}{4} c_0^{-4/3}$ with $\gamma = \left(\frac{2}{3\pi}\right)^{2/3}$, as proven by [66]. More concerns regarding the cutoff parameter can be found in the appendix I or in [55] and [62].

Second, the matrix \mathbf{X} is filled by picking each entry X_{ji} independently from the binary alphabet $\{0, 1\}$ according to $\mathbb{P}[X_{ji} = 1] = p_i$. The independence of the entries X_{ji} holds only in the second step, since two overall random bits X_{ji} and $X_{j'i}$ of the same column are positively correlated since both of them have a higher probability to be 1 if p_i is large.

The most famous distribution and also the most applied throughout this thesis is the *arcsine distribution* introduced in [114] by Tardos. Its corresponding continuous probability density function f is defined within the interval $[t, 1-t]$, with t close to zero. It is symmetric around 0.5 and heavily biased towards values close to the limits of $[t, 1-t]$, as can be seen in the left graphic of figure 2. This is motivated by the marking assumption explained in subsection 4.4. It is the only restriction on the colluders attack model and it is more likely to apply to these columns with a high bias. This choice of the distribution of p_i is used to show that the colluders' attack model only has a minor effect on their chance to be caught. For the binary case the probability density function is defined as

Definition 18 (The Tardos probability density distribution function)

$$f(p) = \frac{1}{2 \arcsin(1-2t)} \frac{1}{\sqrt{p(1-p)}}, \quad p \in [t, 1-t]. \quad (5)$$

4.5.2 Fingerprint generation with discrete bias distribution

This subsection describes an alternative fingerprint matrix generation by means of discrete versions of the arcsine distribution mentioned in the afore subsection. This type of fingerprint generation will be relevant in chapters 7.3 and 7.4 and are used for instance in the approaches by [43], [84] and recently [65]. In the latter, Laarhoven and de Weger proved that for very large collusions ($c_0 \rightarrow \infty$), the results for the discrete distribution tend to those for the continuous distribution, what is adumbrated in figure 3.

Instead of selecting the m bias values p_i as for the Tardos fingerprint generation, for the discrete generation these are selected according to (discrete) supporting points, represented by finite random variables \mathcal{P} , with $\Pr[\mathcal{P} = p] = q = \Pr[\mathcal{P} = 1-p]$. Note that the value q is the

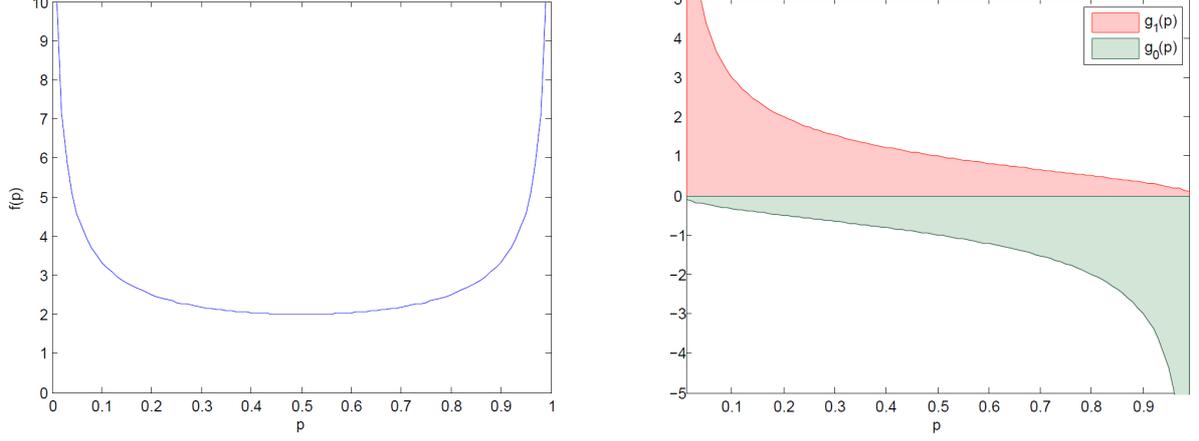


Figure 2: Tardos probability density function and score functions g_1 and g_0 , according to [88]

equivalent to the value $f(p)$ that describes the occurrences of the probability p in the fingerprint matrix \mathbf{X} for the Tardos fingerprint generation. The number of those random variables is prescribed by the number of expected colluders c_0 . Hence we get $(p, q) \in \mathcal{P}_{c_0}$.

According to *Nuida et al.* [85] the distributions are related to *Gauss-Legendre* distributions. For $T \geq 1$, let

$$L_T(x) = \frac{1}{2^T T!} \left(\frac{d}{dx} \right)^T (x^2 - 1)^T$$

be the T -th normalized Legendre polynomial with $L_T(1) = 1$. This polynomial has T simple roots that can be taken as supporting points:

$$(p, q)_T = \left(\frac{x+1}{2}, \frac{2}{(1-x^2)^{3/2} L_T'(x)^2} \right)$$

for $L_T(x) = 0$ [85].

But as the polynomials $L_T(x)$ get more and more complex for increasing T , a method to approximate this distribution is presented by *Laarhoven* and *de Weger* in [65]:

Let $x_{1,T} < \dots < x_{T,T}$ denote the T roots of the polynomial $L_T(x)$ and let $p_{k,T} = (x_{k,T} + 1)/2$ and $q_{k,T} = 2/((1-x_{k,T}^2)^{3/2} L_T'(x_{k,T})^2)$ denote the corresponding values from the distribution $(p_{k,T}, q_{k,T}) \in \mathcal{P}_{2T}$. *Laarhoven* and *de Weger* show in [65] that $p_{k,T} = \sin^2\left(\frac{\pi k}{2T}\right) + o(1)$ and $q_{k,T} = 1/T + o(1/c)$ for $T \rightarrow \infty$.

Therefore, instead of computing the roots of the more and more complex polynomial $L_T(x)$, one can approximate the values $(p_{k,T}, q_{k,T})$ by computing

$$\hat{p}_{k,T} = \sin^2\left(\frac{4k-1}{8T+4}\pi\right) \quad \text{and} \quad \hat{q}_{k,T} = \frac{1}{T}$$

for $1 \leq k \leq T$. This gives the following fingerprint generation process:

1. **Initialization:** Let $m = d_m c_0 \ln(1/\varepsilon_1)$ denote the code length and set $T = \lceil c_0/2 \rceil$. First select m bias values p_i , for $i = 1, \dots, m$, by drawing k for each value uniformly at random from $\{1, \dots, T\}$ and setting $p_i = \sin^2\left(\frac{4k-1}{8T+4}\pi\right)$.

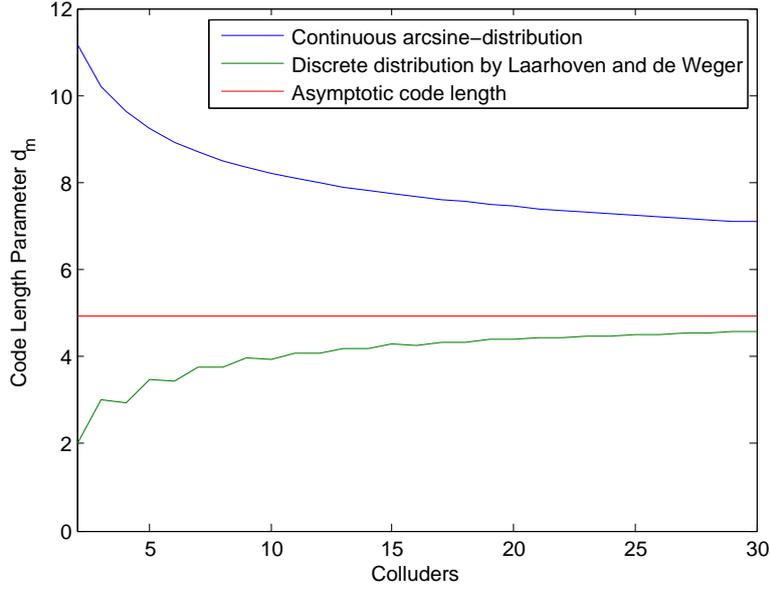


Figure 3: Approximated code length parameter d_m for the continuous arcsine distribution [114] and the discrete distribution by [65] with the symmetric score function [121], taken from [88]

2. **Fingerprint generation:** Each entry $X_{j,i}$ of the fingerprint matrix is selected independently from the binary alphabet with $\Pr[X_{j,i} = 1] = p_i$.

Figure 3 shows the code length parameter for the continuous arcsine-distribution and for the discrete distribution by *Laarhoven and de Weger* [65]. In both cases, the tracing algorithm from *Škorić et al.* [121] (see below) is assumed.

4.5.3 Symmetric Tardos tracing algorithm

When the distributor receives an unauthorized copy and the fingerprint detection process outputs the potentially manipulated fingerprint y , he initializes the fingerprint tracing algorithm. In case he chooses the symmetric *Tardos* tracing algorithm by *Škorić et al.* [121], he first creates the single accusation score S_j between each users' fingerprint X_j and the attacked fingerprint y .

The corresponding single accusation score according to (12) is defined as follows.

Definition 19 (Symmetric Tardos accusation score) For an arbitrary user j , with accusation functions g_1 and g_0 defined as

$$g_1(p) := \sqrt{\frac{1-p}{p}} \quad g_0(p) := -\sqrt{\frac{p}{1-p}},$$

the symmetric Tardos accusation score is calculated as

$$S_j = \sum_{i=1}^m \varphi(y_i; X_j(i)) = \sum_{i=1}^m \delta_{y_i, X_{ji}} g_1(p_{y_i}^{(i)}) + [1 - \delta_{y_i, X_{ji}}] g_0(p_{y_i}^{(i)}). \quad (6)$$

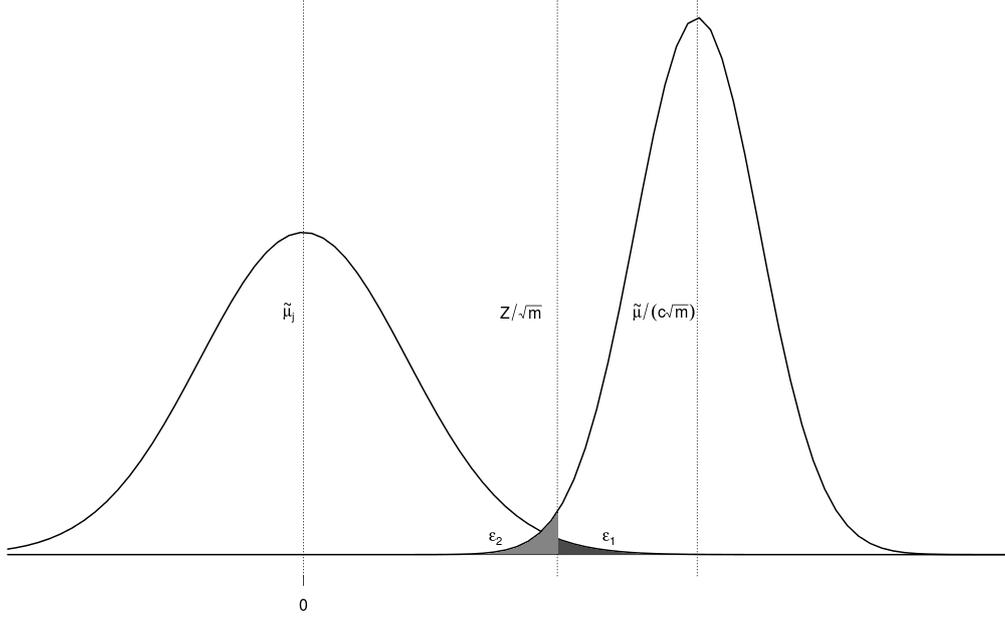


Figure 4: Normalized distribution of the accusation scores in [121]: innocents' scores (left), colluders' scores (right).

where $\delta_{y_i, X_{ji}}$ denotes the Kronecker Delta and $p_{y_i}^{(i)}$ stands for the probability of the symbol in y_i within the entries of the fingerprint matrix X .

The accusation score functions are depicted in figure 2.

To determine, whether a fingerprint is suspicious or not a threshold decision is applied. This means, the distributor suspects a user j to have partaken in the collusion, if the corresponding accusation score S_j exceeds a predefined threshold Z ,

$$\forall j \in \{1, \dots, n\} : S_j > Z \Rightarrow j \in \mathbf{T},$$

where \mathbf{T} denotes the output set of the tracing algorithm, i.e. the set of fingerprints that are suspected of having partaken in the collusion.

The threshold Z depends on the chosen maximum number of colluders c_0 and of the chosen rate for ϵ_1 ,

$$Z \leq d_z c_0 \ln \frac{1}{\epsilon_1}, \quad (7)$$

with a threshold parameter d_z depending on the chosen fingerprint tracing algorithm. For the symmetric *Tardos* tracing algorithm, d_z is set as $\sigma \cdot \pi$. Note that the standard deviation of the expected accusation scores of innocent users σ is assumed to be 1 for the symmetric *Tardos* scheme [121] as well as for the original *Tardos* [114] and most of its derivatives e.g. [86, 21, 73]. Figure 4 depicts the score distribution for the symmetric *Tardos* scheme [121]. Note that the score distributions and threshold Z have been normalized. The figure also illustrates the understanding of the error bounds ϵ_1 and ϵ_2 .

4.6 Sophisticated fingerprinting attacks

A group of malicious customers can execute different collusion attacks in order to avoid detection by the tracing algorithm. Due to the marking assumption by [23], see section 4.4, we assume the colluders are only able to modify detectable positions, i.e. positions where their watermarked media copies differ from each other due to the individual watermark information embedded.

4.6.1 Stateless attack strategies

Apart from that, a typical assumption model found in literature within the marking assumption is the location independence of the attack strategy, in other words, a memoryless collusion channel.

Definition 20 (Stateless attack model) *In an attack model that is location independent, the not-necessarily deterministic output y_i of the position i only depends on the symbols $X_{j_1,i}, \dots, X_{j_c,i}$ of the colluder fingerprints X_{j_1}, \dots, X_{j_c} of that particular position i . It is independent of other positions or any other output y_k , with $k \neq i$. This attack model is defined as stateless attack.*

The reason for the assumption that the colluders restrict themselves to a stateless attack strategy is that the fingerprint generation as well as the tracing algorithm works completely position-symmetric, so it might be disadvantageous for the attackers to deviate from this symmetry. Moreover, it significantly simplifies the proofs for the security of the corresponding fingerprinting scheme. Also, *Moulin* shows in [76] that enforcing this restriction does not change the fingerprint capacity asymptotically, which means for very large collusion sizes.

Let $C = \{j_1, \dots, j_c\} \subseteq \{1, \dots, n\}$ denote a coalition of size c with corresponding fingerprints X_{j_1}, \dots, X_{j_c} . Be X_C the colluder's fingerprint matrix with entries $[X_C]_{,ji} = [X_{j,i}]_{j \in C, 1 \leq i \leq m}$. The colluders may create a forged fingerprint $y = \rho(X_C)$ according to a (possibly non-deterministic) strategy ρ with the constraint that it holds $y_i = X_{j_1,i}$ if $X_{j_1,i} = \dots = X_{j_c,i}$, according to the marking assumption. In stateless attack strategies, the strategy ρ does not depend on the column index i , thus, the different symbols of the forged fingerprint y are generated column by column by $y_i = \rho(X_{C,i})$, where $X_{C,i}$ denotes the i -th column of the colluder-fingerprint matrix X_C .

In literature, it is assumed that the colluders want to participate equally in the forgery. There is no hope in identifying an idle colluder. Hence, we assume that the strategy is invariant under permutation of the colluder identities.

Let b_i be the number of ones in column $X_{C,i}$, with $0 \leq b_i \leq c$. Then the attack strategy ρ can be parameterized by a set of probabilities $\theta = (\theta_{b_i})_{0 \leq b_i \leq c}$ with $\Pr[y_i = 1] = \theta_{b_i}$. The marking assumption enforces $\theta_0 = 0$ and $\theta_c = 1$.

- **Interleaving attack:** For the interleaving attack, an index $k \in C$ is selected uniformly at random and for the symbol position the corresponding colluder-fingerprint value is taken, $y_i = X_{k,i}$. Using the above notation, the interleaving attack can be parameterized as $\theta_{b_i} = b_i/c$. In [54] *Huang* and *Moulin* showed that the interleaving attack is the strongest attack in an asymptotic sense, i.e. if the number of colluders tends to infinity.

- **Minority vote attack:** For the minority vote attack, the colluders choose the least common symbol for that position. In case of a tie, a symbol is selected uniformly at random. The attack can be parameterized as

$$\theta_{b_i} = \begin{cases} 1 & \text{if } 0 < b_i < \frac{1}{2}c \text{ or } b_i = 1 \\ \frac{1}{2} & \text{if } b_i = \frac{1}{2}c \\ 0 & \text{if } b_i = 0 \text{ or } \frac{1}{2}c < b_i < c. \end{cases}$$

- **Majority vote attack:** Analogously to the minority vote attack, here the colluders output the most frequent symbol. In case of a tie, a symbol is selected uniformly at random. The attack can be parameterized as

$$\theta_{b_i} = \begin{cases} 1 & \text{if } b_i > \frac{1}{2}c \\ \frac{1}{2} & \text{if } b_i = \frac{1}{2}c \\ 0 & \text{if } b_i < \frac{1}{2}c. \end{cases}$$

- **All-1 attack:** The colluders output the symbol '1' at each detectable position. The attack can be parameterized as

$$\theta_{b_i} = \begin{cases} 1 & \text{if } b_i > 0 \\ 0 & \text{if } b_i = 0. \end{cases}$$

The **All-0 attack** works analogously, but only that the colluder output a '0' at each detectable position. The bias generation as well as the score function is symmetric, therefore there is no real difference between the all-1 and the all-0 attack.

Note that this attack is solely listed for reason of completeness, in practice the colluders have no knowledge about which positions of the media content belongs to which fingerprint position, nor can they guess the corresponding symbol. Thus, this attack is only interesting in an information theoretic sense and for other application scenarios such as group testing, e.g. [72], and will not play a role in the further procedure of this thesis.

- **Coin flip or random attack:** At each detectable position, the colluders 'flip a coin' to decide whether to output a zero or a one. The attack can be parameterized as

$$\theta_{b_i} = \begin{cases} 1 & \text{if } b_i = c \\ \frac{1}{2} & \text{if } 0 < b_i < c \\ 0 & \text{if } b_i = 0. \end{cases}$$

Note that in case the collusion consists of only two attackers, all attack strategies are substituted to the coin flip attack or random guess attack.

- **Minimal mutual information:** In [41] *Furon* and *Pérez-Freire* presented the worst case attack from an information theoretical point of view. They attempt to find the collusion strategy that minimizes the mutual information between the forged fingerprint y and the

Table 1: Attack strategies that minimize the mutual information between the forged fingerprint and colluders' fingerprints (cf. [41, Table II]).

c_0	$\theta^* = (\theta_0, \dots, \theta_c)$
2	(0, 0.5, 1)
3	(0, 0.652, 0.348, 1)
4	(0, 0.488, 0.5, 0.512, 1)
5	(0, 0.594, 0.000, 1.000, 0.406, 1)
6	(0, 0.503, 0.175, 0.500, 0.825, 0.497, 1)
7	(0, 0.492, 0.000, 0.899, 0.101, 1.000, 0.508, 1)
8	(0, 0.471, 0.000, 0.689, 0.500, 0.310, 1.000, 0.529, 1)
9	(0, 0.440, 0.000, 0.698, 0.230, 0.770, 0.302, 1.000, 0.560, 1)

colluder-fingerprints. They define the achievable rate R against a given collusion attack θ as:

$$\begin{aligned} R(\theta) &= \mathbb{E}_p[I(Y;X|P=p)] \\ &= \mathbb{E}_p[H(Y|P=p)] - \mathbb{E}_p[H(Y|X,P=p)] \end{aligned}$$

with $I(Y;X|P=p)$ denoting the conditional mutual information and $H(Y|P=p)$ refers to the conditional entropy.

To make a clear distinction between innocent users and colluders gets harder, if the mutual information is comparably small. This argumentation in general is independent of the tracing algorithm. If R is zero, the forged fingerprint y is independent of the colluder-fingerprints and it is not possible at all to distinguish between innocent user and colluder. *Furon et al.* found the stateless attack strategy that minimizes the mutual information for collusion sizes $c < 10$ using numerical optimization. The corresponding attack strategies θ are listed in table 1. Note that the attack strategy by *Furon et al.* is not necessarily the worst case attack strategy for the specific fingerprint generation, tracing algorithm or collusion size. Different attack strategies might be more efficient due to a suboptimal construction of the corresponding tracing algorithm.

4.6.2 Stateful attack strategies

As the colluders do not know fingerprint messages of innocent users, they cannot raise a specific innocent-fingerprint's score to increase his chance to get accused. Instead, they aim to stay undetected by trying to lower their own accusation scores. They succeed, if all colluders get an accusation score lower than the threshold Z .

As mentioned in the afore section, a typical assumption in literature is the location independence of the attack strategy. Due to definition 20, the *stateless* attack model does not depend on the index i nor on the symbols for the other detectable positions or the selected output for other positions.

However, in the real world the colluders are not restricted to this assumption. Instead they can apply attack strategies for which each symbol is chosen also with respect to the symbols observed on all other detectable positions.

Definition 21 (Stateful Attack Model) In an attack model that is location dependent, in order to choose the symbol y_i of the current position i , the colluders may use the information about the symbols y_k , $k = 1, \dots, m$ taken for the other positions $k \neq i$ as well. This attack model is referred to as *stateful attack*.

To generalize this observation, we need the subsequent definitions and statements. Define the accusation score as $S_j = S'_j + S''_j$, where the score computed over the non-detectable positions is denoted as S'_j , and S''_j represents the score computed over the detectable positions. The value S'_j is equal for all colluder-fingerprints and cannot be changed due to the marking assumption. Hence, the optimal attack strategy to avoid getting accused can be defined as in Definition 22.

Definition 22 (Optimal Attack Strategy) An attack strategy is referred to as *optimal*, if it outputs a manipulated fingerprint y^* for a collusion C with arbitrary $j \in C$, that minimizes the probability that the largest accusation score of the colluder-fingerprints exceeds the threshold Z .

$$\Pr[S_{\max} > Z] = \Pr[S''_{\max} > Z - S'_j]$$

As the actual column probability value p_i of the fingerprint matrix \mathbf{X} is unknown to the colluders, they cannot compute S''_{\max} exactly. However, for instance using the maximum-likelihood method, the colluders may guess each p_i from the observed symbols. Let b''_i denote the number of observed 1's at position i . The maximum-likelihood method results to the estimation $\hat{p}_i = b''_i/c$ and

$$\hat{S}''_j = \sum_{i \text{ detectable}} g(X_{j,i}, y_i, \hat{p}_i).$$

The value

$$\hat{y}^* = \arg \min_y (\max_{j \in C} \hat{S}''_j)$$

therefore appears to be a reasonable candidate to minimize $\mathbb{E}[S''_{\max}]$.

We define the stateful attack strategy that aims to find valuable approximations of \hat{y}^* in an efficient way as *Greedy Attack*, as it applies a *greedy* strategy to generate y .

- **Greedy Attack:** For this stateful attack strategy the colluders aim to minimize their accusation scores, i.e. to minimize $\mathbb{E}[S''_{\max}]$. More precisely, they approximate the column probabilities p_i , $i = 1, \dots, m$, of the fingerprint matrix \mathbf{X} by analyzing their colluder matrix X_C . With the approximated values for p , they compute the accusation scores of all colluders up to position $i - 1$. For the selection of the symbol in the current position i of the manipulated fingerprint y , they choose whether a '1' or a '0' according to the symbol that decreases the accusation score that was the highest up to that position.

For a more detailed description of the greedy attack and for an additional stateful attack strategy, we refer the reader to section B in the Appendix.

4.7 Summary

Chapter 4 describes the general structure of a fingerprinting scheme for watermark applications and classifies the different types of fingerprinting schemes. For the type that is most important

in both literature and this thesis, namely for the static probabilistic fingerprinting schemes, the three processes of each scheme that are especial for fingerprinting, i.e. the fingerprint generation process, the collusion channel and the tracing algorithm, are explained in more detail. Providing more insight into these processes, we showed the example of the famous *Tardos* code [114, 121, 64] and completed the section listing common sophisticated collusion attack strategies as well as proposing the new model of stateful attack strategies, that have to be considered in some practical applications.

In summary, with the choice of the eligible fingerprinting parameters, the number of users n , the maximum number of colluders to be resistant against c_0 , and the desired upper bound on the False Positive error ε_1 , the code length m for the chosen fingerprinting code can be calculated and thus the dimensions of the fingerprint matrix \mathbf{X} are fixed. With it the content provider is able to generate the fingerprints using a continuous distribution or a discrete distribution.

Note that in many practical applications the watermarking payload prescribes the limit of the code length m . Therefore, the interaction of the above mentioned parameters can be reorganized to serve the payload requirements. The matching of the varying requirements for the different application scenarios is devoted to the subsequent chapter 5.

5 Requirement analysis of fingerprinting codes in the considered application scenarios

In this chapter we identify the requirements for the collusion secure fingerprinting codes according to the application scenarios described in chapter 3. These requirements will form the basis for the algorithms and fingerprinting schemes described in chapters 6 through 9. Especially important is the conclusion, that the different scenarios necessitate varying fingerprinting schemes. It will become clear that the different requirements for the different scenarios typically cannot be satisfied by one single fingerprinting scheme.

In order to describe and evaluate these requirements, the parameters and definitions are taken that were introduced in the afore chapter 4. The main significance for the classification of application scenarios to the appropriate fingerprinting schemes is devoted to the parameters c_0 , n , and M , the maximum number of colluders the fingerprinting code should be resistant against, the number of users to be accommodated in the system, and the maximum payload that is provided by the media copy, respectively. Additional analysis is performed for requirements on the probability that the scheme misfires in its accusation decision, i.e. the probabilities that the errors exceed their acceptable bounds ε_1 and ε_2 . Although this thesis is not engaged in legal affairs, a comparison to the usability of DNA may provide a directive for the selections of ε_1 and ε_2 . According to *Lindsey et al.* in [69], assuming a Gaussian distribution, 10 persons (including the responsible perpetrator) out of 10 millions would be associated to the same DNA-profile. Due to test-inaccuratenesses, one assumes 100 false positive errors on average for 10 million persons tested, which equals a false positive rate of 10^{-5} . Therefore, for our fingerprinting solutions proposed in the upcoming chapters we assume bounds on the error rates that are chosen close to this rate, in order to increase the acceptance of fingerprinting codes (at least) as clear indication when it comes before court.

An especial focus is laid on the maximum number of colluders the fingerprinting code should be resistant against c_0 . This parameter, its impact on the code length respectively, and thereby its choice of fingerprinting code characterizes the actual application scenario. In summary, in the further procedure of the thesis, we decide between three magnitudes to classify the potential of a collusion size.

Small: A collusion size indicated as *small* is mainly restricted to up to three colluders.

Medium: A *medium* sized collusion consists of at least four through up to 15 colluders.

Large: A *large* collusion comprises more than 15 colluders.

Note that this separation is not obligatory, but it assists classifying the results of this thesis in the next chapters. Though we make a clear distinction between *small collusion sizes*, *medium collusion sizes* and *large collusion sizes*, the switchovers oftentimes are blurring. In the following we distinguish between parameter values classified as *small*, *medium* or *large* not only for c_0 but also for the other parameters n and M , in order to classify the requirements for the current scenario. Therefore, the distinctions can be seen as a general statement of the requirements of the actual application scenario and the corresponding chosen fingerprinting scheme.

5.1 Fingerprinting codes for promotional and pre-release purposes

The promotional and pre-release application scenario as defined in chapter 3.3 requests the following requirements for a suitable fingerprinting scheme. The given payload is the crucial part. Since the promotional application scenario uniquely depends on its corresponding

end-customer scenario, the payload that is available generally stays the same (given the same requirements on the basic watermarking scheme). The expected number of colluding customers c_0 is relatively small since the overall number of users that receive promotional material is small as well. For example, according to an analysis by *The New York Times*¹, the jury that votes for the *Oscar's* contains about 6000 members. However, not every member gets to vote for every category, which means in this scenario the number of users n is clearly limited by 6000.

Moreover, the receivers of promotional material are typically not acting anonymously, at least the copyright holding party or company has a legitimately signed contract about the restrictions dealing with their copyrighted material. This reduces the potential of large collusions and, on the contrary, improves the probability that the evidence brought by fingerprinting codes holds before court. For this reason, the promotional application scenario in particular, in which users that were bound by a clear contract collude and get accused, provides solid ground for fingerprinting codes.

Besides the probabilistic fingerprinting schemes that will be considered in this thesis, especially the promotional scenario provides high potential for a deterministic fingerprinting scheme, i.e. a scheme that is able to mathematically prove the guiltiness of colluders. That is, if the identity of the user that was legitimately bound by a contract is assigned clearly to the mathematically proven guilty fingerprint, the judges would have a very strong argument to sue this user guilty. However, the applicability of a deterministic scheme obviously depends as well on the payload that is available for the actual media cover.

For the promotional application scenarios as classified in chapter 3.3 the following requirements arise regarding the maximal number of colluders c_0 that can be expected, and regarding the number of fingerprints needed, i.e. the number of users n , as well as the payload M that typically is provided by the current scenario.

- **Maximum expected collusion size c_0 :** For the applicants of fingerprinting schemes a maximum number of colluders the scheme is provable resistant against is desired to be arbitrary high. Fingerprinting schemes satisfying this requirement are inter alia [121] [7] or [86]. However, the strong dependency on its applicability requires a more realistic choice for c_0 . The potential for large collusions is assumed to be relatively small for all the promotional media types, i.e. musical promotion, audio books, Ebooks, movie/TV promotion as well as the pre-release phase for video games. It seems plausible that journalists for music publishers do not want to expose themselves in a collusion. The threat of a collusion consisting of more than two journalists therefore is negligible. The same holds for audio books and movie/TV promotion, since there are cases known in which common transaction watermarks already lead to damning evidence, as already posted in chapter 3.3. Ebook promotion until now is not established to an amount for which it is obligatory to resist against large collusions, a proper watermarking basis combined with a fingerprinting scheme resistant against up to two or three colluders appears appropriate. For the different phases in the life-cycle of a video game as postulated in [12], see figure 5 of section 5.4, also the potential of large collusions changes. In the first phases of the life-cycle of a video game there is a potential of at least small collusions. During the later testing phases, named as *Beta* and *Master*, that precede the official release of a game, sometimes large numbers of testing persons have access to the game. Depending on the game and the environment there is a risk of a collusion that can be sorted into a medium sized collusion.

¹ <http://mediadecoder.blogs.nytimes.com/2011/05/23/electronic-voting-comes-to-the-oscars-finally>

- **Number of fingerprints/users n :** As most fingerprinting schemes require that the number of fingerprints n has to be chosen in advance in order to generate the fingerprint matrix \mathbf{X} , see chapter 4, the distributor needs to have a clear position about the number of copies that need to be protected by fingerprints. For the promotional application scenario this number is typically determined from the first. Compared to the corresponding end-customer scenario, only a relatively small number of selected persons will receive a media copy. In analogy to the expected number of colluders c_0 , solely for the pre-release phases of the video game scenario the parameter value for n exceeds the range of being *small*.
- **Available payload M :** The crucial part for all fingerprinting schemes is the given payload M of the media copy that enforces very strong restrictions on the fingerprints and leads to compromises between the participating fingerprinting parameters. The payload for the media types within the promotional application scenario is by nature equal or similar to the end-customer product that it advertises. More precisely, for musical promotion the corresponding song, whatever the format, provides only very limited payload. Using state of the art watermarking algorithms, only few hundred bits can be embedded into two through four minutes of audio material, [110]. Consequently, the payload for audio tracks already is a challenge for common transaction watermarking methods, a fortiori it is even more challenging for fingerprinting schemes. Audio books, as discussed above, are rarely applied for promotional purposes, however, they generally provide a magnitude of the payload provided in musical promotion. Therefore, the potential for embedding is classified as medium amount of payload. For Ebooks there is a controversy. Though there exists algorithms for embedding watermarks in the content-text, the more common method is embedding watermarks into the images that are included as well as into the *images* of the cover/title pages. Thus, depending on the cover medium where the watermark information is embedded, the available payload is rather small, yet in general larger than for the musical promotion scenario. The promotion of movies and TV productions, generally provides a larger amount of payload due to the nature of including video as well as audio content and also because the average time range exceeds the typical time range for musical promotion by far. However, efficiently watermarking video content remains challenging, what limits the amount of payload that can be embedded. In summary, for the promotion of movies and TV productions we postulate a medium amount of payload that is available for the embedding of fingerprints. On the contrary to the media types listed above, the potentially available payload in boxed video games is huge. As discussed in chapter 3, this is due to its different nature, i.e. because a video game consists not of a single media file of one type, but it consists of a huge number of files per different media component. Video games thus provide the maximum payload of all discussed media types and consequently they are suitable for the longest fingerprints that can be realistically embedded.
- **Acceptable upper bounds ε_1 and ε_2 for the error probabilities:** The error probabilities are desired to be as small as possible and thus the corresponding bounds should approximate zero. However, as already discussed, this is generally not realistic. Finding appropriate selections for the values of ε_1 and ε_2 is challenging. The applicant of fingerprinting codes, i.e. the distributor, has a strong desire to catch at least one colluder, which means he tends to choose a value for ε_2 that is very small. However, when it comes before court, the bound on the probability that the accusation is fraudulent, i.e. that the probability bound for accusing innocent customers is relatively high, gets especially important. Therefore, in general the selection for ε_1 is sharper than for ε_2 . According to Škorić *et al.*

Table 2: Classification of the parameters payload, number of users and maximal expected collusion size in the promotional application scenario

Promo. application scenario	Music	Audio books	Ebooks	Movie/TV	Video games
Max. expected colluders c_0	small	small	small	small	small/medium
Number of users n	small	small	small	small	small/medium
Available payload M	small	medium	medium	medium	large

[121] an upper bound on the probability that on average half of the colluders get accused is acceptable ($\varepsilon_2 = 0.5$), whereas the bound on the probability of accusing a specific innocent ought to be significantly lower. Oftentimes a value complying to $\varepsilon_1 \approx 1/n$, i.e. one out of the number of users, is assumed as realistic to get submitted to court.

In summary, the promotional application scenario provides solid legal ground and the potential of collusion attacks is mainly restricted to small collusions. Consequently, for the most media types within this scenario we expect a small number of users, i.e a small number of fingerprints and an analogously small maximal expected collusion size at mostly small available payload, partly medium sized and solely for the video game scenario there is large payload available. Table 2 mirrors the results of this discussion for the parameters c_0 , n and M .

5.2 Fingerprinting codes for online shops

A lot of what was just discussed for some media types in the promotional application scenario can be transferred to the online shop scenario as well. However, as this is the classical end-customer scenario, in general the requirements on the fingerprinting parameters increase. In the following we list the requirements on the fingerprinting parameters maximum expected collusion size c_0 , number of users n , and the available amount of payload M , as well as the error bounds ε_1 and ε_2 according to the media types that serve the online shop application scenario.

- **Maximum expected collusion size c_0 :** Compared to the promotional application scenario discussed in the section above, the online shop end-customer scenario has a higher potential for large collusions. For example, in the field of music distribution, whatever the format, it can be assumed that groups of dishonest customers that are aware of the watermark protection collude in order to avoid being traced back. However, it seems unlikely that very large collusions are formed, because not only is the benefit for the colluded material, e.g. a song, relatively small, but colluding on a wide scale is also a matter of trust. The rationale for the value of c_0 in music distribution therefore is to be resistant against medium sized collusions. The same holds for the distribution of audio books, images and video clips. There might be the potential of medium sized collusions, but large collusions seem very unlikely to appear. Also, in the distribution of Ebooks the potential of large collusions and even of medium sized collusion seems very low. However, this assumption is only theoretic, as there is not sufficient experience with attacks on watermarked Ebooks until now, to give a clear statement. Streaming media however, though obviously strongly depending on the content to be streamed, raises the potential for large collusions. Especially for events that are live-streamed and interest a huge audience, it appears conceivable that large collusions are formed. Even more in the scenario of video game

Table 3: Classification of the parameters payload M , number of users n and maximal expected collusion size c_0 in the Online shop application scenario

	Music	Audio books	Ebooks	Images	Videos	Media streams	Video games
c_0	medium	medium	small	medium	medium	large	large
n	large	large	large	large	large	large	large
M	small	medium	medium	small	medium	large	large

distribution, since the market is known for active piracy and professional groups attacking the game protection, and because a typical boxed video game promises higher benefit compared to one music track for instance, the distributors reasonably desire protection against a large maximum expected collusion size.

- **Number of fingerprints/users n :** The online shop distribution generally is designed for unlimited and ongoing number of customers, which means that a choice regarding the value for n for the fingerprint generation process is very speculative. The overall opinion is to expect a large number of users, meaning a large number of fingerprints to be generated in advance. This number is strongly depending on the content, no matter if in music distribution, audio books, Ebooks, images, video clips, streaming media or video game distribution.
- **Available payload M :** The available payload M for the online shop distribution scenario obviously is equal to the payload given in the corresponding promotional application scenario. Therefore, the available payload for music distribution is very small. The payload for audio books and Ebooks stays medium sized and for the video game distribution via online shop the available payload remains huge. In addition, the given payload for images typically is significantly small, in general even smaller than the payload for music. Images typically provide payload of no more than 100 bits, making the application of fingerprints almost impossible. The potentially available payload for video clips is sorted as a medium amount, because of the possibility to embed fingerprints into both audio track and video track. For streaming media it again strongly depends on what content is streamed, but in general assuming a large amount of available payload seems reasonably.
- **Acceptable upper bounds ε_1 and ε_2 for the error probabilities:** The required bounds on the error probabilities in the online shop scenario remain as strict as for the promotional application scenario (or even stronger). Thus, the parameters chosen as $\varepsilon_1 \approx 1/n$ and $\varepsilon_2 = 0.5$ appear feasible.

In summary, the Online shop application scenario requires similar parameter values for the error bounds ε_1 and ε_2 as the promotional application scenario. Also the available payload M for most media types, apart from the video game scenario where sometimes only parts of the game are handed out, is the same as well. This is contrary to the required parameter values for the number of users n and for the maximum expected collusion size c_0 . Both strongly depend on the distribution content, though in general there exist the potential of medium sized collusions. For the video game scenario there even is the potential for large collusions, thus requiring large values for c_0 . The number of users n for all media types of the Online shop distribution scenario can be classified as large. Table 3 depicts the results on that discussion.

5.3 Fingerprinting codes for digital cinema

The digital cinema application scenario as identified in chapter 3.6 differs from the end-customer model alike the online shop application scenario as well as from the promotional application scenario. Here, the customers that receive watermarked media content play the role of intermediaries and utilize the watermarked content to further offer a service to their (end-)customers. Though the model of the digital cinema application scenario covers more than solely the name giving digital cinema scenario, as already discussed in chapter 3.6, we will restrict ourselves to this scenario in the further procedure of this thesis. This is due to the main relevance for fingerprints in the scenario of digital cinemas compared to the other scenarios within this model, for which the potential for collusion attack is negligibly small.

In chapter 3.6 we explained the need for watermarking both audio and video in a movie, i.e. watermarking audio is more robust compared to video watermarking but the audio track per se is of less importance than the video track and therefore it might get exchanged. In the following we will assume there is no distinction of whether audio or video or both is capable to get watermarked, but we will assume there is only one media type contained in a movie, and thus there is only one value for each of the fingerprinting parameters c_0 , n , ε_1 and ε_2 as well as for the available payload M . Remember that in the attack scenario for digital cinemas it is not the cinema itself that plays the role of an attacker, yet this role is played by people visiting the movie. However, as these generally remain anonymous, and so do the criminals that re-record the movie, the fingerprint intends to solely identify the specific cinema where the unauthorized re-recording happened. For the collusion potential this means, that re-recorded movies from different cinemas are compared and attacked. Consequently, tracing back to the cinemas that provided material for a collusion attacked movie does not automatically mean finding the responsible criminals. However it might help preventing the next attack.

- **Maximum expected collusion size c_0 :** The requirements for the digital cinema application scenario regarding the expected number of colluders depends on the movie content, on the number of cinemas that receive a copy, and on the country where it is screened. The benefit and the possibilities to conduct a collusion attack are rather high, whereas the need for a large collusions might yield more risk and effort than needed, as the fingerprints only carries the potential to identify the correct cinemas but not the attackers themselves. Thus, a parameter value for c_0 that is of medium size appears feasible.
- **Number of fingerprints/users n :** Though the number of visitors of a movie definitively is rated as very high, the number of cinemas that screen that particular movie is much smaller. Indeed, in the scenario of fingerprinting for digital cinema application, the number of fingerprints n is represented by the number of cinemas not by the number of visitors. However, again strongly depending on the content, also the number of cinemas can be classified as a large value, ergo n is required to be large.
- **Available payload M :** Due to its nature, the available payload for embedding fingerprints into movies is equal to the closely related scenario for movies and TV productions in the promotional application scenario as discussed in section 5.1. Consequently, the movie in the digital cinema application scenario provides available payload M of medium sized amount.
- **Acceptable upper bounds ε_1 and ε_2 for the error probabilities:** The acceptable upper bound for the probability of a false positive error ε_1 in the scenario for digital cinemas is less strict compared to the other scenarios described before. This is because the *innocent*

that gets falsely accused is not a person that would be made liable for the unauthorized action, as it is in the other scenarios, but the *innocents* that get falsely accused are the cinemas where the movie was screened but no (related) unauthorized action took place. The cinemas this way or another are not made liable for the re-recording, but inter alia they can be enforced to improve their controls and security mechanisms. The damage caused by a false alarm is not as intense as in the other scenarios. It remains lower than the upper bound ε_2 for the probability of not finding a responsible cinema, i.e. a false negative error. In summary, the restrictions on the values for ε_1 and ε_2 generally could be lowered, though for simplification purposes, and as their influence is minor compared to e.g. c_0 , we keep the same values as in the sections above, ergo $\varepsilon_1 = 1/n$ and $\varepsilon_2 = 0.5$.

Overall, for the digital cinema application scenario we assume the following requirements for the fingerprinting parameters. At a large number of cinemas, which means a large number n of fingerprints to be generated, we expect the potential of at most medium sized collusions c_0 and adopt the values for the bounds on the error probabilities ε_1 and ε_2 from the sections above, though they also could be lowered. Finally, we work with the assumption of a given payload M of medium size.

5.4 Fingerprinting codes for video games

Contrary to classical media, where the product is represented by one single file and thus only this file ought to be protected, in the video game scenario, the product consists of large sets of files of various media types, among others textures, sound files or 3D models, as depicted in figure 1 of chapter 3. Moreover, the process of development of one single boxed game may last several years and throughout the different phases of the development, as depicted in figure 5, we suggest different watermarking layers for its protection, see [12].

Therefore, the video game application scenario brings new opportunities but also new challenges. In the following we list the corresponding requirements regarding the fingerprinting parameters c_0 , n , ε_1 and ε_2 . We will also state an assumption regarding the available payload M . Note that though a video game consists of a magnitude of different files of different types, we will consider it as a whole, i.e. one single product.

- **Maximum expected collusion size c_0 :** As already discussed in chapter 3.6 and partly in section 5.1, the potential of large collusions is clearly existent. The so called release groups are professionally organized groups of hackers who intend to break the protection of video games. Moreover, each of these groups try to break this protection as soon as possible in order to boast being the first. For the application of fingerprints, i.e. for the publishers, this means collusion security against a large maximum number of colluders c_0 is vital for their business.
- **Number of fingerprints/users n :** The number of users n obviously is depending on the content, but for the AAA titles this number is huge, typically it is significantly larger than for the other scenarios discussed above. For example *Ubisoft's* role play game *Assassin's Creed 4, Black Flag*, released in November 2013, shipped 11 million copies until May 2014², though it does not belong to the 10 most shipped games of that year according to the famous video games statistics magazine *VGChartz*³. Note that the given sales number sums the sales numbers of all platforms.

² <http://www.ign.com/articles/2014/05/15/assassins-creed-4-ships-11-million-copies>

³ <http://www.vgchartz.com/yearly/2013/Global/>

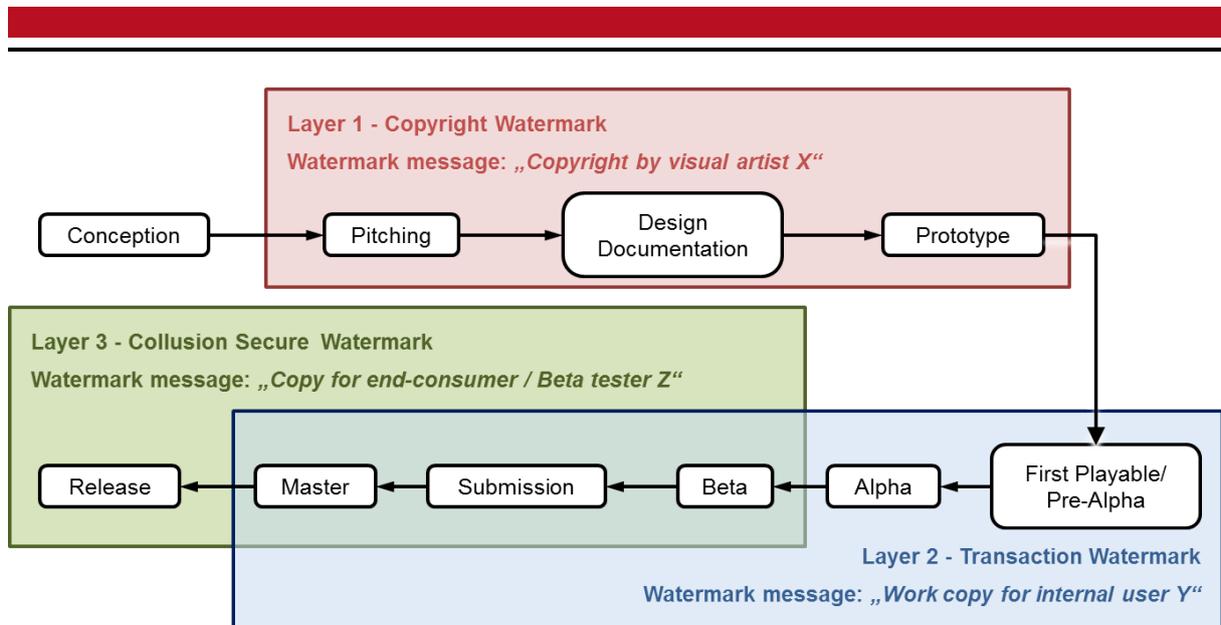


Figure 5: Different watermarking layers during the various phases of the life-cycle of a boxed video game according to [12]

- **Available payload M :** The different nature of video games compared to classical multimedia such as music or images offers an especial huge amount of payload. If the watermarking basis is capable to embed information into all possible components, the available payload M is immense. However, a proper watermarking basis, though theoretically possible, in practice still lacks of sophisticated tools that are able to handle all media types and also manage the very varying sizes within one type. For the process regarding watermarking video games we refer the interested reader to e.g. [12], here we assume the payload provided by the watermarking algorithm as given. The total amount of payload M that is available for fingerprint embedding therefore is massive, thus M is classified as large.
- **Acceptable upper bounds ε_1 and ε_2 for the error probabilities:** The video game community generally is very critical regarding protection mechanisms. Therefore, in case it gets public that an innocent user is accused of having partaken in a collusion attack, i.e. in case there is a false positive error, the community hardly will forgive the fingerprinting/watermarking system and accept it as appropriate protection. Therefore, the upper bound ε_1 on the probability of a false positive error needs to be very close to zero. Since the number of users for this scenario, as described above, is comparably large, the value for ε_1 , that is $1/n$, appears acceptably small. The desired upper bound ε_2 for the probability that no colluder gets accused, ought to be small as well. However, for simplification purposes and to stay comparable to the other scenarios, we keep the value as $\varepsilon_2 = 0.5$.

The video game application scenario is the first scenario to allow embedding of fingerprinting codes on a wide scale. Therefore, taking the same parameter values for the error bounds ε_1 and ε_2 as before, the fingerprinting parameters c_0 and n are both considered as *large*.

5.5 Fingerprinting codes for databases/hashtables

This work proposes a new method that, in case the list is found, allows tracing back illegal transmission of blacklists and whitelists by individualizing each list. This means, for example, if several providers of anti-virus programs purchased such a list, the list would be individualized

for every provider. Thereby, if one of these providers distributed his list and it appeared at an illegal location, it would be possible to trace it back to him. The result could be that this provider gets informed to improve his protection methods.

This application scenario is very different from the other scenarios above that exclusively apply watermarking of media content. In this scenario, the fingerprint positions are represented by dummy hash entries. Though this changes the amount of data, i.e. the overall number of hash entries, a few thousands additional entries in a set of several hundred thousands hashes presumably will not raise suspicion. However, also contrary to the media watermarking scenarios, where deleting detected fingerprint positions is strongly affecting the content, deleting hashes in general is possible and will not render the hashtable meaningless. Therefore, the requirements on the appropriate fingerprinting schemes are different as well. Moreover, there is no related or state of the art work that acted on this scenario before. Consequently, the application of fingerprinting codes in hash tables is yet widely unexplored and clear argumentation regarding the requirements is result of our observation.

In the following we list the expected requirements on the fingerprinting parameters c_0 , n , ε_1 , ε_2 as well as on the potentially available payload M .

- **Maximum expected collusion size c_0 :** The potential for large collusions appears to be rather small. In this field of distributing sensible data, the union of a large number of those that receive an individualized hashtable, in order to break the protection is not realistic. Therefore the required maximum number of attackers to be resistant against c_0 is at least of *medium* size.
- **Number of fingerprints/users n :** The number of different receivers of hash tables in the scenario of blacklists and whitelists as described is not comparable to the number that is needed in an end-customer watermarking scenario. In general the number of individualized hash tables, i.e. the number of fingerprints is relatively *small*, presumably as small as in the promotional application scenario.
- **Available payload M :** The payload for the embedding of fingerprints cannot be stated clearly. This is because in this scenario there is no media watermarking basis that embeds the fingerprints, more precisely it is not yet clarified how the fingerprints ought to be embedded. Therefore, we anticipate the corresponding fingerprint embedding technique, that is embedding dummy hash entries as fingerprint positions, and argue that the available payload corresponds to the number of hashes that are contained in the hashtable. An increment of hashtable entries of about 5% due to fingerprint embedding will rather not raise suspicion. Assuming hash tables of several hundred thousands entries, the available payload for the embedding of fingerprints into hash tables definitively is *large*.
- **Acceptable upper bounds for the error probabilities ε_1 and ε_2 :** Due to the sensitivity of the content in this scenario, the error probabilities ought to remain very low. However, as there is no persuasive precedent, we have to apply the lowest upper bounds for the error probabilities suitable for the corresponding fingerprinting scheme that is applied. Therefore, for a first approach, it appears most reasonable, to keep the same values as for the other scenarios. Thus we start with $\varepsilon_1 = 1/n$ and $\varepsilon_2 \approx 0.5$ to be comparable to the other scenarios.

5.6 Summary

In this chapter we elaborated the required parameter values for the corresponding fingerprinting schemes in the different application scenarios.

Table 4: Identified requirements on the fingerprinting codes for the different application scenarios

Application scenario	Promotional	Online shop	Digital cinema	Video games	Hash table
Max. expected collusion size c_0	<i>small</i>	<i>medium</i>	<i>medium</i>	<i>large</i>	<i>medium</i>
Number of fingerprints n	<i>small</i>	<i>large</i>	<i>large</i>	<i>large</i>	<i>small</i>
Available payload M	<i>medium</i>	<i>medium</i>	<i>medium</i>	<i>large</i>	<i>large</i>
False positive error bound ε_1	$1/n$	$1/n$	$1/n$	$1/n$	$1/n$
False negative error bound ε_2	$1/2$	$1/2$	$1/2$	$1/2$	$1/2$

We classified the attributes *small collusion*, *medium sized collusion* and *large collusion*. The resulting requirements that were identified are listed in table 4. Recall that the promotional application scenario in section 5.1 as well as the online shop application scenario in section 5.2 contain different media types. For this reason, in order to fill the columns *Promotional* and *Online shop* in table 4, we excluded the video game scenario in each case and also the field of streaming media regarding the online shop application scenario. The corresponding requirements are shifted to the column for *Video games* in general.

In general we reason that there are scenarios in which for all fingerprinting parameters only *small* values are required. Moreover, in none of the scenarios listed above there was postulated a *large* value for the maximum expected collusion size c_0 where at the same time the corresponding available payload M was assumed to be *small*. This argumentation in general raises hope for the possibility of applying fingerprinting codes for the different application scenarios and different parameters. However, the discrepancies in the requirements enforce the application of different fingerprinting schemes for different application scenarios, different in the sense, that they are adjusted or adapted to the targeted scenario. With the fingerprinting schemes found in literature, a matching of suitable parameters to the actual need is rarely evident, nor is it elaborated in a way that potential applicants are satisfactorily instructed how to adjust these parameters. To the best of our knowledge, these matching of requirements from given scenarios to the best selection of fingerprinting schemes has not been provided before. Therefore, we are the first to focus fingerprinting codes with respect on their (watermarking) application scenario. In the upcoming chapters the resulting fingerprinting schemes that match the corresponding application scenarios will be introduced.

A general requirement that we exact is the adaption of the fingerprinting schemes to the available payload. Unless the code length m of the fingerprints does not exceed the corresponding value for M , the fingerprinting codes cannot be applied at all and further discussion would be meaningless. The following summary lists the properties of fingerprinting codes that are required for the identified application scenarios.

-
- **Zero false positive providing fingerprinting schemes:** Especially eligible for all fingerprinting schemes that are applied in practice and that intend to provide legal evidences before court, are deterministic fingerprinting schemes that provably do not accuse innocent users. As already discussed, this requirement is hardly enforceable. However for very small values of the maximum expected collusion size, i.e. $c_0 = 2$, chapter 6.1 proposes our patented solution for a deterministic fingerprinting scheme that additionally provides the shortest code lengths, [94] and [95].
 - **Short fingerprinting codes for a small user basis resistant against a small number of colluders:** Especially in the promotional application scenario in section 5.1 the required values for the fingerprinting parameters are relatively small and so is the payload given by the most media types. In order to provide collusion resistance at all, the distributor ought to take fingerprinting schemes that, though providing resistance solely against small maximum collusion sizes, are sufficiently compact to be embedded at all. Therefore, we propose our solution of a simple but efficient *3-secure* fingerprinting scheme in chapter 6.2.
 - **Flexible fingerprinting codes for easy adaption of changing requirements of the fingerprinting parameters and varying payload sizes:** Whether for small values of the fingerprinting parameters or for larger values, the distributors need the possibility to adjust the fingerprinting scheme according to their current need. Especially in the Online shop scenario of section 5.2, in which the distributor is not restricted to one product, e.g. one single song, but typically provides a large portfolio of products, easy adaption of the current needed parameters for the current product is vital. Our solutions optimized regarding these requirements are introduced in chapters 6.2, 7.3 and 7.4, as well as in chapter 8.
 - **Optimized fingerprinting codes for practical parameter choices and realistic error rates:** In general the applicant of fingerprinting schemes obviously desires to apply the optimal fingerprinting code. However, since for the practical parameter values there does not exist the *optimal* code, the best selection of available fingerprint schemes, i.e. best selection of fingerprinting generation as well as tracing algorithm, is important. Even more because the error rates proposed in literature are received for theoretical parameters and dependencies within the scheme that in practice are not necessarily realistic. This requirement holds true for all application scenarios, but is especially important in the online shop application scenario 5.2 as well as in the digital cinema application scenario 5.3. Therefore, our solutions that are introduced in chapter 7.4 and in chapter 8 clearly expose the best selection of existing schemes and its optimal adaption to the practical needs.
 - **Performance optimized fingerprinting codes for large parameter values in real world scenarios:** In addition to the general requirements of optimized fingerprinting codes regarding its parameters, some scenarios necessitate to consider the performance of the tracing algorithms as well. This especially holds for the scenario of watermarking video games 5.4 but is also true for the scenarios considering online shops 5.2, e.g. for streaming media, and for digital cinema applications 5.3. That is because the complexity grows exponential with increasing values for the fingerprinting parameters. Chapter 8.1 introduces our solution that first satisfies these requirements in a joint tracing scenario while retaining a practicably low computational complexity. Additionally, the solution proposed in chapter 7.1 also provides a means to manage the complexity for large values of the fingerprinting parameters.

-
- **A general adaption of fingerprinting codes for hashtables:** The scenario of fingerprinting hashtables is special. For example, contrary to the other scenarios, there is no watermarking basis needed to embed the fingerprints, they are simply added to the hashtable via dummy hash entries. Though, on first sight this has no effect on the fingerprinting parameters, it opens the possibility of a new kind of attack strategy, that is deleting (fingerprint) positions. However, the appropriate fingerprinting scheme has to be optimal regarding small values for the maximum number of colluders c_0 and number of users n , but gets the opportunity of a large value for the available payload M . A corresponding solution is not existent at all, therefore, we are the first to propose an approach that allows applying fingerprinting codes for hashtable distribution. This approach is introduced in chapter 9.

6 Short fingerprinting codes against small collusions

As identified in chapter 3, we need appropriate fingerprinting codes suitable to be embedded as transaction watermark messages into different types of media of different formats.

In this chapter we describe two different fingerprinting schemes that are optimized regarding the properties required for those application scenarios described in chapter 5 for which the available payload is very limited, i.e. solely very short fingerprinting codes can be embedded. The approaches differ largely regarding fingerprint generation, tracing algorithm and error diagnosing. They also differ in the maximum number of colluders to be resistant against, as the one approach is fixed 2-secure, and the other approach is fixed 3-secure.

According to the nature of fingerprinting codes as described in chapter 4, in order to generate short fingerprinting codes, the maximum expected number of colluders c_0 ought to be selected as small. This especially matches the requirements for the promotional application scenario as identified in chapter 5.1. Hence, the two fingerprinting codes introduced in this chapter are particularly targeted on this scenario. Both methods provide suitable short code lengths m for the required maximum collusion sizes c_0 at negligible small error bounds ε_1 and ε_2 .

For some media types within the online shop scenario described in 5.2, the limitations on the available payload M solely allow embedding of short fingerprints as well. In order to provide collusion security at all, the fingerprinting codes against small collusion sizes introduced in the two upcoming sections might be applied in those scenarios as well.

6.1 2-Secure fingerprinting

In this section we present our approach published in [94] and patented under [95]. It is collusion-secure against two colluders and especially suits the requirements for the promotional application scenario as described in chapter 5.1. In chapter 3 it was pointed out that very short code lengths of sometimes less than 100 bits are required. At the same time a deterministic statement regarding the validity of the assumed colluders increases the value of the scheme. The approach that is introduced in the following provides some deterministic properties in the sense that there is a probability for a false negative error, but false alarms cannot occur, i.e. $FP = \varepsilon_1 = 0$. At the same time the code length is comparably low. It is capable to stay below $m = 100$ bits for realistic parameters. This is achieved by requiring special conditions for the fingerprint matrix \mathbf{X} . That is the Hamming Distances between each pair of fingerprints of \mathbf{X} ought to be within a prescribed interval. This property is utilized by a specific parent pair searching algorithm that is applied as tracing algorithm, closely related to the approach by *Nuida et al.* [83].

Fingerprinting codes that focus on the resistance against two colluders in order to provide short code lengths have already been published, e.g. [26], [39], [80]. These are labelled as *2-secure codes*. However, these approaches whether appear as not sufficiently compact ([26], [39]), or they do not provide the security level that is desired for this scenario ([26], [39], [80]).

The 2-secure fingerprinting scheme we present in the following is the only approach presented in this thesis, that does not belong to the classes of *Tardos Codes* as described in chapter 4. That is because it utilizes neither a probabilistic fingerprint generation nor a corresponding tracing algorithm based on scoring functions. Though different from the other schemes, it still contains the especial fingerprinting processes (as described in 4.1).

1. **Fingerprint generation process:** With the choice of the parameter n representing the number of fingerprints needed, the distributor estimates the appropriate code length m and starts the generation of the fingerprints, in a way that all fingerprints have a pairwise Hamming Distance to each other within a prescribed interval. The fingerprint generation is described in more detail in the following construction section 6.1.2.
2. **Collusion channel:** For this approach the collusion channel is restricted to the attack by two malicious customers. Therefore, all stateless attack strategies as described in chapter 4 do not differ from each other. For this reason we abstain from a wide discussion about the attack strategies but consider them during the evaluation section 6.1.3.
3. **Tracing algorithm:** For the tracing algorithm we adopted the idea of applying identifying parent property (IPP) codes. Our approach represents a parent pair searching algorithm closely related to the approach by *Nuida et al.* [83]. The complete tracing algorithm is detailed in the corresponding paragraph of section 6.1.2.

The fingerprints can be embedded and detected as watermark messages by means of transaction watermarking algorithms.

6.1.1 Preliminaries

In chapter 5.1 we pointed out, that on the one hand the strong requirements regarding the payload limitations for many media types presuppose code lengths of few hundred bits. On the other hand a distinct deterministic statement regarding the validity of the output of the tracing

algorithm is desirable. Both issues especially hold for the promotional application scenario. The deterministic statement can be provided solely by *deterministic* fingerprinting schemes, for which the provided code lengths exceed the practical range by far. All approaches providing comparably short code lengths belong to the class of *probabilistic* fingerprinting schemes, which means the provided security is not deterministic but estimated via certain error probabilities, see chapter 4.

For the generation of the fingerprint matrix the number of fingerprints, i.e. the number of users n , must be determined in advance. The length of the codes m is independent of the false positive error bound ε_1 and maximum collusion size c_0 , that are zero and two respectively, by construction. The number of fingerprints n appears as the essential parameter to receive both, m and the upper bound of the false negative error probability ε_2 . This is contrary to the *Tardos Codes* and its derivatives.

As distance metric between two arbitrary rows of the fingerprint matrix \mathbf{X} , we select the *Hamming Distance* or L_1 norm. Therefore, the maximum distance per fingerprint position is 1, the distance of two fingerprints X_j and $X_{j'}$ is computed as the total number of bit positions having a different symbol ('0' or '1'). It is denoted as $HD(X_j, X_{j'})$. Examples of distance metrics for codes using several alphabets are for example listed in [49].

The proposed approach applies a specific modification of a parent pair searching algorithm by *Nuida et al.* in [80] in order to trace the colluders. They proposed short 2-secure fingerprinting codes by means of a searching algorithm belonging to the classes of IPP Codes, see chapter 3.1, called *parent pair search*. That is exploring every constellation of a pair of fingerprints, who might be responsible for the possibly manipulated fingerprint y . However, checking every constellation of (pairs of) fingerprints for its possibility of having partaken in the collusion attack is computationally very demanding. Therefore, in [79] *Nuida* optimizes the computationally expensive approach of [80] applying a pre-search based on Hamming Distances before a modification of his searching algorithm. Thereby he reduces the number of fingerprints that come into consideration for the cost of a slightly modified parent pair search, whereby the effort is reduced essentially.

The parent pair search of [80] roughly works as follows. It analyzes every bit position of a pair of fingerprints $X_j, X_{j'}$ if it is possible that those fingerprints could possibly be the *parents* of the attacked fingerprint y , i.e. $y_i \in \{X_j, X_{j'}\}$. The algorithm returns 'YES' if every bit could have been created in collaboration of both, otherwise it returns 'NO'. Every pair of fingerprints for which the algorithm returns 'YES' is reasonably denoted as *possible colluder pair*.

6.1.2 Construction

This subsection details the construction of the approach. It is divided into the fingerprint generation process and the tracing algorithm.

Fingerprint generation process

The idea is to require specific preconditions on the binary fingerprints, to get a stronger dependence between each. With this preconditions we include additional information to the fingerprint matrix \mathbf{X} that later allows a precise accusation decision in the process of the tracing algorithm.

In the $n \times m$ fingerprint matrix \mathbf{X} , where m represents the number of bits, e.g. the code length, and n stands for the number of users to be listed in the system, each row represents a fingerprint for a particular user, see chapter 4. The matrix \mathbf{X} is created as follows.

An efficient tracing algorithm that uses Hamming Distances as measurement for the affiliation of y , was originally suggested by Hagiwara *et al.* [44]. They applied Hamming Distances as some sort of accusation scores, see definition 12 in chapter 4.3. However, the corresponding fingerprint matrix \mathbf{X} is generated via a coin flipping algorithm.

In order to supply more information from the fingerprint generation process for the tracing algorithm, the preceding idea was to prescribe a distinct Hamming Distance between each row of \mathbf{X} to every other. With this property the detection of a collusion and identifying the colluders via a scoring function that uses Hamming Distances would be trivial. However, it can be shown that the length m of the rows of a matrix becomes impractically large when the matrix is created such that the Hamming Distances for all possible pairs of fingerprints is equally large [49].

For this reason we allow the Hamming Distance between two fingerprints to be in a predefined interval $[HD_{\min}, HD_{\max}]$. The fingerprint matrix \mathbf{X} has to fulfill the *Hamming Distance condition*:

Definition 23 (Hamming Distance condition) *The Hamming Distance condition describes the following requirements for the fingerprint matrix \mathbf{X} :*

- For every possible pair of fingerprints $(X_j, X_{j'})$ of \mathbf{X} , with $j \neq j'$, it must hold: $HD(X_j, X_{j'}) \in [HD_{\min}, HD_{\max}]$.
- For HD_{\min} and HD_{\max} it must hold:
 - i) $HD_{\min} > \frac{1}{2}HD_{\max}$
 - ii) $HD_{\min} \leq \frac{1}{2}m$
 - iii) $HD_{\min} + HD_{\max} = m$

The parameters HD_{\min} and HD_{\max} represent the lowest Hamming Distance value allowed and the largest Hamming Distance value, respectively.

In order to generate sufficient fingerprints suiting this condition, the inequalities of definition 23 can be exhausted. This means setting HD_{\min} as approximately $1/3$ of the fingerprint length m and HD_{\max} slightly smaller than $2/3$ of m .

One way to create the matrix \mathbf{X} is to first creating a larger matrix X' of the dimensions $n' \times m$, where $n' \gg n$ (empirical experiments yield $n' \approx 10 \cdot n$ as appropriate). This larger matrix can be filled for example applying a coin-flipping algorithm. To receive a feasible matrix \mathbf{X} out of X' , a simple deletion of all rows of X' that do not meet the Hamming Distance condition introduced above is proceeded. Note that this pragmatic matrix creation is only one possible way. Every generation of \mathbf{X} that satisfies the Hamming Distance condition 23 is feasible.

Parent pair tracing algorithm

The tracing algorithm is separated into the following steps. For a more detailed description please see [94] and the appendix C.

- First the Hamming distance between every fingerprint X_j , with $j = 1, \dots, n$, of the fingerprint matrix \mathbf{X} to the possibly manipulated fingerprint y is calculated.

- Those fingerprints with a Hamming Distance smaller than or equal to $\frac{1}{2}HD_{\max}$ are associated to one distinct group of fingerprints. The corresponding group is denoted as G_1 .

$$G_1 := \{X_j; HD(X_j, y) \leq \frac{1}{2}HD_{\max} \forall j \in \{1, \dots, n\}\}$$

- Those fingerprints with a Hamming Distance below or exactly $\frac{3}{4}HD_{\max}$ and (at the same time) exceeding $\frac{1}{2}HD_{\max}$ are associated to another group of fingerprints. The corresponding group is denoted as G_2 .

$$G_2 := \{X_j; \frac{1}{2}HD_{\max} < HD(X_j, y) \leq \frac{3}{4}HD_{\max} \forall j \in \{1, \dots, n\}\}$$

- Those fingerprints with a Hamming Distance exceeding $\frac{3}{4} \cdot HD_{\max}$ are sorted out. The corresponding fingerprints are no longer considered.
- The parent pair search of [80] is applied to all combinations of pairs of fingerprints possible in group G_1 .
- The parent pair search of [80] is applied to all combinations of pairs of fingerprints possible, for which one fingerprint is associated to group G_1 and the other is associated to G_2 .

After these steps there might be several pairs of fingerprints labelled as *possible colluder pair*. These are considered for the final decision by means of theorem 1.

- Theorem 1**
1. *If there is associated exactly one pair of fingerprints as possible colluder pair; and at the same time there is no fingerprint with a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$, this possible colluder pair is the colluder-fingerprints.*
 2. *If there is associated more than one pair of fingerprints as possible colluder pair and one fingerprint appears in all possible colluder pairs and at the same time there is no different fingerprint with a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$, this fingerprint is a colluder-fingerprint.*
 3. *If no pair of fingerprints is assigned as possible colluder pair; the fingerprint with the minimum Hamming Distance to y is a colluder fingerprint.*

This means, if one of the cases of theorem 1 applies, the corresponding fingerprint (pair) yields the output of the tracing algorithm. The proof of theorem 1 can be found in the appendix C.

The sorting into the two groups G_1 and G_2 is done, because the Hamming Distance of at least one of the colluder-fingerprints, denoted as X_{C_i} with $i \in \{1, 2\}$, to y is at most $1/2HD_{\max}$. This is due to the marking assumption as explained in chapter 4.4. Therefore with out loss of generality it holds $HD(X_{C_1}, y) \leq HD(X_{C_1}, X_{C_2})/2$. This means at least one colluder-fingerprint is always in the process of group G_1 . Moreover, if none of the fingerprints is closer to y than $\frac{1}{4}HD_{\max}$, it is assured that both colluder-fingerprints show up in $G_1 \cup G_2$ (see appendix C). In this way it is guaranteed that if exactly one pair of fingerprints appears as possible colluder pair, it must be the colluder-fingerprints. If the parent pair outputs a YES more frequently than once, we have to clarify if one fingerprint is contained in every possible colluder pair. In this case the corresponding fingerprint is output as colluder-fingerprint, otherwise no fingerprint is output.

Table 5: Required code length and its errors of our approach in comparison to *Nuida's* in [79]

fingerprints n	code length m	Nuida [79]		Our approach	
		FP	FN	FP	FN
10^2	61	10^{-4}	$> 10^{-2}$	0	0.0018
	71	10^{-5}	10^{-2}	0	0.0044
	91	10^{-7}	10^{-3}	0	0.0004
10^3	74	10^{-4}	$> 10^{-2}$	0	0.033
	83	10^{-5}	$> 10^{-2}$	0	0.011
	93	10^{-6}	10^{-2}	0	0.0044
	102	10^{-7}	$> 10^{-3}$	0	0.0021

6.1.3 Evaluation

To evaluate the results on code length and error probabilities, the approach is compared to *Nuida's* approach of [79], which, to our knowledge, provided the best results so far for the discussed parameter selection. The empirical results are shown in table 5. For identical code lengths and for similar values for the average number of the event of a false negative error (FN) as published in [79], we obtain a false positive error (FP) of *zero*, compared to FP error bounds of 10^{-4} through 10^{-7} by [79].

The particular code lengths in table 5 were selected in order to be comparable to the empirical results by *Nuida*. As postulated in 5.1, the goal was to reduce the false positive error rate at possibly no cost in the other parameters such as false negative error rate and code length, compared to the state of the art. Please note that a code length of 61 is not the shortest possible with related bound on the false positive error $\varepsilon_1 = 0$ and a very small bound on the false negative error ε_2 . For 10^2 users we are able to provide similar error bounds for a code length of 57.

In summary, to the zero false positive error rate, there comes a very low false negative error rate for shortest code lengths obtained so far. Code lengths of less than 100 bits still providing optimal error bounds are the particular attributes of this approach. The requirements regarding the error bounds as postulated in 5.1 are satisfied. More results can be seen in the appendix C.

The approach already is implemented into Fraunhofer SIT's anti-piracy audio watermarking solution. All tests for which the approach was applied with the working audio watermarking scheme confirm the analytical and empirical results gained so far. Moreover, the approach was successfully tested with Fraunhofer SIT's image watermark solution. Within practical tests with real images, also additional forensic possibilities to further reduce the false negative error were exposed [101]. As these are strongly entangled with the watermarking algorithm, we will not examine this here.

6.2 3-Secure Fingerprinting

In this section we describe a 3-secure fingerprinting scheme that is built in a way that it can be adjusted to the actual need of the applicant easily, but still provides high security and shortest code lengths against at most three colluders [92]. With respect to chapter 5, this is especially utile for the promotional application scenario as discussed in 3.3 but can be reasonably applied in other scenarios that are prone to small collusions as well.

The code is very flexible and consciously kept simple to be adaptive and fast in order to stay applicable for all demands of potential appliers. In lieu of making use of the expectation values of the accusation scores solely for the computation of the error bounds alike the symmetric *Tardos* fingerprinting code described in chapter 4.5, in this approach we utilize the actual accusation scores induced by the current case of collusion attack, for a first decision about which kind of attack strategy the colluders may have used. Depending on this decision, the tracing algorithm decides between two different tracing strategies. Both are based on discarding as many fingerprints as possible and only consider the most suspicious for a final decision.

There already exist *3-secure codes*, e.g. [82], [99], but the computational effort and the complexity of the scheme are impractically high with respect to some media scenarios. Our approach achieves at least equal results regarding security and code length at much less effort. All state of the art approaches try to stay independent of the attack strategy. This is a requirement that in some way prevents of better achievable results, i.e. shorter code lengths, faster tracing algorithms, sharper error bounds, etc, but is indispensable to keep a small attack surface for the colluders.

Our approach proposed in the following consists of a slightly modified *Tardos* fingerprint generation process, see chapter 4.5, and a twofold tracing algorithm. By means of the actual accusation scores by the current case of collusion attack, we decide between two tracing strategies that are constructed to counter whether a minority vote (or alike) strategy or majority vote (or alike) strategy.

The parameter set required for the scheme are the code length m , the number of fingerprints n and the upper bounds on the error probabilities, analogously to the *Tardos Codes*.

6.2.1 Preliminaries

The fingerprint generation for the *Tardos Code* [114] follows a certain distribution to circumvent the dependency of different attack strategies, see chapter 4.5. It is structured in a way that the mean value of the accusation scores for innocent-fingerprints is balanced to zero with a variance of one. The scores for the colluder-fingerprints will have a distinctive positive value.

This approach in some way resembles *Tardos'* fingerprint distribution [114] in order to reduce the dependency of the attack strategies. However, for a short code that is resistant against three colluders, an applicable code length is too small to hinder the impact of the conducted attack. This means, the accusation scores achieved by the proposed 3-secure fingerprinting scheme still significantly differ for different attack strategies.

Compared to the other possible stateless attack strategies, see chapter 4.6, that match the marking assumption explained in chapter 4.4, the (Hamming) distance between y and a colluder-fingerprint is smallest in case of a minority vote attack. It is largest in case of a majority vote attack.

With respect to the (Hamming) distance, the other attacks are sorted in between the minority vote and the majority vote attack. Note that this does not reflect the results achieved by means of probabilistic accusation scores in general.

For this reason, the proposed 3-secure fingerprinting scheme consists of two different tracing algorithms. Depending on the attack strategy, each tracing algorithm achieves different results. The one tracing algorithm achieves competitive error rates for a small code lengths and still is fast enough for practical applications in case of an attack tending to the minority vote attack. But for an attack tending to the majority vote attack the resulting error probabilities are fairly suitable.

The other tracing strategy provides similar proper results yet for the majority vote attack strategy, but the results are unsatisfactory for the minority vote attack.

To get the optimum of both tracing strategies, the attacked fingerprint y is evaluated in order to receive knowledge about the attack that was applied to generate y .

As the attack strategy, whether minority vote or majority vote or else, directly influences the symbols in y , the score of y to itself reveals information about the attack strategy. This is reasoned by the following trick: We calculate the accusation score of y to itself. Note that this represents the case in which there is only one malicious customer re-distributing his media copy containing his fingerprint not manipulated at all. Obviously, only positive amounts are added for this accusation score. This means the properties p_i that are used to fill the columns of the fingerprint matrix \mathbf{X} , see chapter 4.5, are the decisive factors for this score. It is likely that the columns of the colluder matrix X_C also follow the probabilities of \mathbf{X} . This means, if p_i is very low in the i th column and thus only few '0's appear in this column, the number of '0's in the corresponding column of the colluder matrix is low as well. Hence, in a minority vote attack, where always the symbol occurring the least is taken to create y , see chapter 4.6, the probabilities p_y for these symbols are low as well. With the inverse proportion of the probabilities to the amount added to the accusation score, see for instance figure 2, we reason a coherence of the score S_y and its collusion attack that created y . In other words: The larger S_y , the more likely it is that the colluders conducted a minority vote attack, and vice versa. With it we are not able to precisely tell the actual strategy, but it tells something about its tendency. Other attacks result in values for S_y somewhere in between the results for minority vote and majority vote attack. To decide between minority tracing or majority tracing, we apply an empirically identified threshold β . Tests showed a success rate for this decision of more than 95 percent. This success rate is not outstandingly high, but in case the decision is wrong, the corresponding fingerprints, and thus the attacked fingerprint y , are by chance favorably generated anyway – due to the probabilities p . This means, also a wrong decision, i.e. the improper tracing strategy, results in a reliable output of the tracing algorithm, see table 24 in the appendix D.

For the process of tracing colluders we need the following descriptions and definitions.

A triple of fingerprints of matrix \mathbf{X} is denoted as (t_1, t_2, t_3) . Be t_{j_i} the symbol of fingerprint t_j in position i . The envelope of a triple of fingerprints (t_1, t_2, t_3) is defined as

$$\epsilon(t_1, t_2, t_3) := \{(\alpha_1, \dots, \alpha_m) \mid \forall i = 1, \dots, m \exists t_{j_i} : \alpha_i = t_{j_i}, j \in \{1, 2, 3\}\}$$

A triple that could possibly have created the attacked fingerprint y is denoted as **colluder triple**.

Definition 24 (Colluder triple) A colluder triple is a triple of fingerprints, t_1, t_2, t_3 , for which the colluded fingerprint y belongs to their envelope $\epsilon(t_1, t_2, t_3)$.

The definition of a colluder triple resembles the idea of the parent pair from section 6.1 expanded for three *parent* fingerprints.

6.2.2 Construction

The following two paragraphs detail the fingerprint generation process and the tracing algorithm respectively.

Fingerprint generation process

The basic fingerprint generation process is based on the *Tardos Codes* [114]. Technically, *Tardos*' choice of the distribution of p_i is used to show that the colluders' attack strategy has only a minor effect on (an exponential average related to) their chance to be caught [114]. Our choice of the distribution of p_i for this approach differs from [114], because we want the attack strategy to have a distinct effect on the code.

The distribution is realized by the probability density function f' which is also symmetric around 0.5 and biased towards values of p_i close to the limits of the interval $[t, 1 - t]$ but additionally with a local maximum at $p = 0.5$.

Definition 25 (Probability density function f')

$$f'(p) = \frac{1}{N} \left[(p(1-p))^{-\frac{1}{2}} + a \cdot (p(1-p))^b \right].$$

Note that the first summand of the right side $(p(1-p))^{-\frac{1}{2}}$ represents the ordinary beta function also taken for the original *Tardos Code* [114]. The second part is an adaption to emphasize the effect of the accusation score of y to itself, see [92]. N denotes the normalization constant ensuring that the integral of the probability density function f' over $p \in (0, 1)$ equals 1. The parameters a and b regulate the curve progression of f' . In figure 6 these are set to $a = 200$ and $b = 4$. These parameter choices approved best in empirical results. They are kept for the evaluation 6.2.3.

Instead of generating the $n \times m$ matrix \mathbf{X} according to the probability density function f' , we create two matrices X_l and X_r , each of size $n \times m/2$ and each individually generated according to f' .

The two $n \times m/2$ matrices put together represent the $n \times m$ fingerprint matrix \mathbf{X} .

3-secure tracing algorithm

The tracing algorithm works the following steps.

- The tracing algorithm starts calculating the accusation scores according to equation (6) in chapter 4.5.

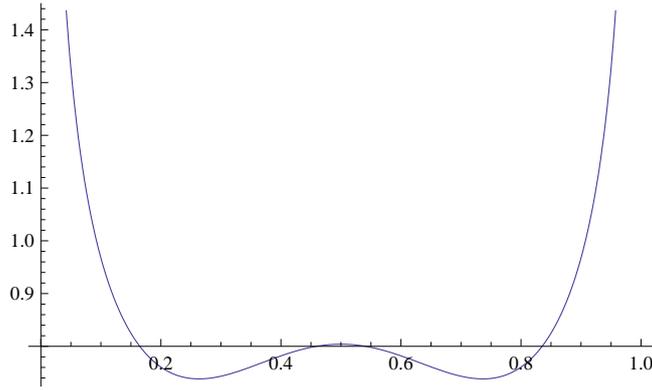


Figure 6: The probability density function f'

- The accusation score of the manipulated fingerprint y to itself is denoted as S_y . It is calculated according to

$$S_y(y, y) = S_y = \sum_{i=1}^m g_1(p_{y_i}^{(i)}).$$

These scores perform as a pre-evaluation of potential colluder-fingerprints.

- The value of S_y decides which tracing strategy to apply. If the value exceeds a certain threshold β , y probably was created in an attack strategy tending to the minority vote attack. The tracing strategy that corresponds to this decision is referred to as *minority tracing*. If S_y stays below β , the tracing strategy denoted as *majority tracing* is applied. The threshold β and the corresponding decision is empirically justified.

Majority tracing:

The majority tracing strategy applies a technique similar to *Meerwald* and *Furon's* joint decoding algorithm [74].

At first we calculate a certain value s , that serves to separate between potential colluder-fingerprints and fingerprints that probably have not partaken in the collusion.

$$s = \left\lfloor \frac{m \cdot S_y}{c_0 \cdot \beta \cdot \ln(n)} \right\rfloor \quad (8)$$

The s fingerprints with the highest scores are sorted into a group G_{sus} of suspicious fingerprints.

The value s in equation (8) depends on the code length m , the number of fingerprints n and of the attack strategy in form of S_y . For very large values of S_y , there is no need for s to be large in order to keep a high security level. Whereas if S_y is close to β , a large value for s is required. This is because, if the decision is tight, the probability that the assumption of the attack strategy was wrong is comparably large. This means, the probability of a colluder-fingerprint not being sorted into the group of suspicious fingerprints increases.

All possible triples of the group G_{sus} of suspicious fingerprints are generated. For each triple is tested if the attacked fingerprint y could possibly belong to the envelope of the current triple $\epsilon(t_1, t_2, t_3)$, if the current triple is a colluder triple according to definition 24 respectively.

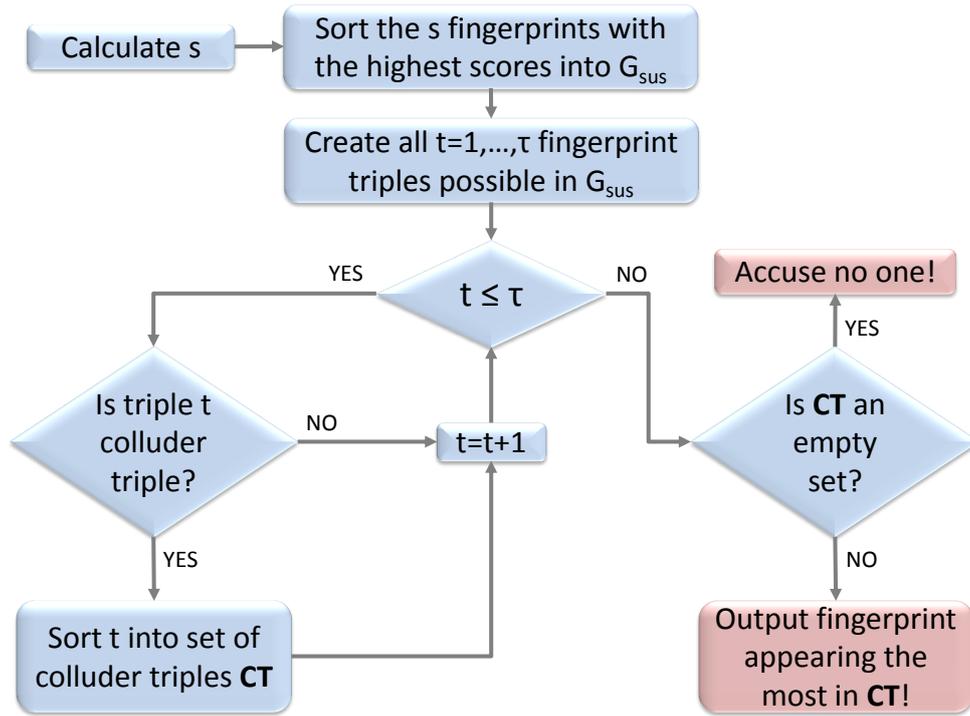


Figure 7: Majority tracing strategy

Finally, the fingerprint that appears most in all colluder triples found is output as being a member of the collusion. A rough description of this tracing strategy is depicted in figure 7.

Note that in this example only one fingerprint is output by the tracing algorithm. Allowing a small decrement regarding security, it is feasible to output the second and/or third most appearing fingerprint as well, provided that these three themselves form a colluder triple.

Minority tracing:

The minority tracing strategy is applied if S_y exceeds the threshold β . This is the case if the Hamming distances of the colluder-fingerprints to y are comparatively large. A tracing strategy analogously to *majority tracing* does not provide sufficient results.

Score calculation: First, y is split into two halves y^l and y^r . For both corresponding halves of each fingerprint X_j the two accusation scores S_j^l and S_j^r are calculated. Both halves are treated independently from each other, in the following we restrict our explanations on the left side, S_j^l , y^l , etc. The right side is proceeded analogously.

Group generation: In case the accusation score S_j^l of the left side of fingerprint X_j exceeds a certain threshold Z^l , X_j is sorted into a group G^l . The threshold Z^l is computed (for example) in dependency of the mean value of all scores of y^l . The choice of Z^l regulates the trade-off between computational complexity and security for the further procedure of the tracing algorithm. For the infrequent case that G^l is empty or consists of only one fingerprint, it is simply filled up to a certain number with the fingerprints that achieve the highest scores. For performance reasons during the next steps, we propose to sort G^l according to the maximum scores in descending order.

Pair building: Next we build all possible pairs of fingerprints out of group G^l . The set of pairs is denoted as PG^l . For every pair of fingerprints in PG^l , we calculate the Hamming Distance

to y^l . In other words, we calculate how many symbols in y^l could possibly be generated by the current pair of fingerprints. The set of pairs PG^l is sorted according to the minimum Hamming distance to y^l in ascending order. If two pairs provide the same Hamming distance, the pair containing the fingerprint with the highest accusation score is favored, because of the proposed sorting order of G^l from the last step.

Triple searching:

1. The first pair within PG^l , referred to as $(X_{j_1}^l, X_{j_2}^l)$, is compared to the first pair $(X_{k_1}^r, X_{k_2}^r)$ of the corresponding set of pairs of the right side PG^r (that is created analogously).
2. Check if there is a fingerprint whose fingerprint halves appear in the first pair of PG^l as well as in the first pair of PG^r , i.e. if there exists an fingerprint index j that appears in both pairs, e.g. (X_j, X_k) and (X_j, X_r) . If this is the case, continue with step 3, otherwise jump to step 4.
3. Check if y belongs to the envelope of the corresponding three fingerprints X_j, X_k, X_r , i.e. check if the triple (X_j, X_k, X_r) is a colluder triple. If this is the case, the algorithm stops and outputs X_j as a colluder-fingerprint. Otherwise continue with step 4.
4. Proceed with step 1 by comparing stepwise the next pairs of PG^l with the next pairs of PG^r .

This way is proceeded for all pairs of fingerprints of both sets until a colluder triple is found or all pairs of PG^l have been considered to all pairs of PG^r .

A rough description of the algorithm is depicted in figure 8. More information can be found in [92] or in the appendix D.

The proposed specification of the algorithm is only one example of a set of different ones, that differ in the chosen thresholds or limitation values within the process. Depending on what purpose the scheme is applied to and which limitations regarding code length, effort, and error bounds are provided, the scheme can be adjusted.

6.2.3 Evaluation

This approach provides a very fast and though effective tracing algorithm. It is simple and easy to manage by construction. This means the scheme may be adjusted to the individual needs of the applicants easily.

To demonstrate the efficiency of our scheme, we selected the parameters $a = 200$ and $b = 4$ for both testbeds of table 7 and table 24.

Table 6 shows the empirical error rates for different code lengths m and different number of fingerprints n . As expected, the larger the code lengths the lower the error rates, and the larger the number of fingerprints the higher the error rates. The chosen values for the parameters m and n are selected in order to show the variability of the scheme.

In table 7 our results are compared to the results of *Nuida* in [82]. Accordingly, the choice of attack and the parameter selection for code lengths and number of fingerprints is adopted from [82]. The table shows the results for a minority vote attack. This makes sense, since for both approaches the minority attack strategy forces the most effort for the tracing algorithm and often is hardest to trace back. Our approach improves the error rates compared to [82].

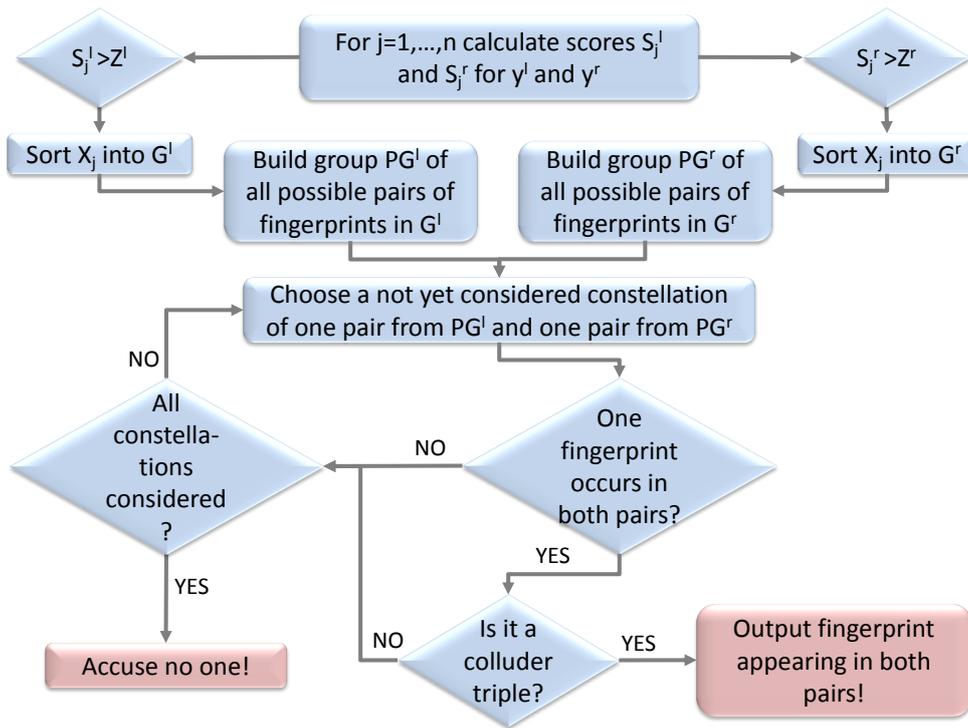


Figure 8: Minority tracing strategy

Table 6: Probability of a FP, i.e. an innocent fingerprint among the output colluder triple, depending on number of users and code length

number of fingerprints n :	10	100	1000	1.000.000
	probability of a false positive			
	100	0.0001	0.00023	0.0038
code length m	250	$1.1 \cdot 10^{-13}$	$4.8 \cdot 10^{-10}$	$5.23 \cdot 10^{-7}$
	500	$3.5 \cdot 10^{-28}$	$1.5 \cdot 10^{-24}$	$1.67 \cdot 10^{-21}$

Table 7: Comparison of the proposed approach to *Nuida* [82]

code length m	no. of fingerprints n	error rate <i>Nuida</i> [82]	error rate our approach [92]
135	100	0.009	$4 \cdot 10^{-5}$
180	1000	0.001	$< 10^{-5}$

More concerns regarding expansion of the algorithm and more tests with different attack strategies proving the comparability of the scheme can be found in the appendix D.

6.3 Summary

In this chapter we proposed two approaches of fingerprinting codes that are resistant against collusions of maximum 2 or 3 colluders and therefore serve as a distinct protection for the scenarios that were classified to provide potential for attacks of small collusions as analyzed in chapter 5. For the scenarios that expect attacks by larger collusions, these approaches are not suitable, as the false negative error rate increases significantly with larger collusions. In general, the false positive error rate is not affected.

false positive error bound: Both approaches intend to reduce the false positive error as small as possible. The 2-secure approach provides an approved zero false positive error rate, and therefore is unique among the probabilistic fingerprinting codes. Note that this approach does not belong to the family of Tardos Codes. The 3-secure approach also provides competitive low false positive error rates.

false negative error bound: Both approaches achieve much lower false negative error rates than assumed to be sufficient for the corresponding scenario.

code length: Reducing the code length is the primary concern for both approaches. Both provide shortest code length for whether 2-secure or 3-secure codes compared to the state of the art.

number of fingerprints: The 2-secure fingerprinting scheme is restricted to a relatively small number of fingerprints provided that the code length is kept to the minimum. This is due to the Hamming Distance condition 23 that constricts the number of fingerprints that can be generated. For a set of 10^3 fingerprints the scheme still is the best choice. To be able to generate a set of 10^4 fingerprints that suffice the Hamming Distance condition, the code length needs to be extended significantly. In this case other approaches in literature are to be favored, e.g. [80], [8]. The 3-secure approach, alike most Tardos Codes, is only negligibly affected by the number of fingerprints.

complexity: Due to its nature of a limited number of fingerprints, the 2-secure tracing algorithm is per se very efficient. The fingerprint generation however can be time consuming. The 3-secure approach is by intention very simple and compared to the state of the art, e.g. [82] and [8], the complexity is very low.

The different parameter sets and more test results regarding the error rates and the code length are arranged in the appendix C and D. The tests prove the comparability and quality of the two approaches.

The 2-secure fingerprinting approach proposed in this chapter in a more general form has been patented. It was submitted in September 2010 and approved in March 2012 [95].

7 Optimized fingerprinting codes against medium sized collusions

In chapter 6 we introduced two approaches to be resistant against two or three colluders. In this chapter we introduce four different approaches capable to resist against larger collusion sizes. For several reasons these are most effective for medium sized collusions. Other publications that intend handling comparable parameter selections are for instance [84], using discrete distributions for the fingerprint generation process, and [73], iteratively outputting colluder fingerprints utilizing these as side information for the next iteration.

Our approaches partially (though simultaneously) take on those ideas, though all are based on the works [114], [121], [64] and [86]. They refinish those state of the art fingerprinting schemes in several ways. They differ in the targeted process, whether with respect to the fingerprint generation process or to the tracing algorithm and also regarding their originator scheme that they add modifications to. All four approaches have in common that they reduce the false positive error compared to the state of the art at minimal cost for the other parameters but for a low increment of complexity.

7.1 Efficient tracing by filtering suspects: Ranking search

This approach generates fingerprints similar to the Tardos codes described in chapter 4.5. Its main contribution is the efficient and adaptive tracing algorithm that employs a ranking search based tracing of suspects.

In literature the fingerprints achieving accusation scores that exceed a certain threshold are either directly output, e.g. [121, 91] or chosen for further analysis, e.g. [81], [75], [40]. The approach introduced here can be handled in both ways. On the one hand it is a self-contained tracing strategy providing a very fast and high accurate colluder tracing directly outputting one or more colluder-fingerprints. On the other hand it acts as a filter selecting the most suspicious fingerprints in order to constrict the number of fingerprints to be considered. This way the effort for adjacent ordinary tracing algorithms is significantly reduced. The approach was published in [93].

The proposed approach is suitable for most fingerprinting schemes utilizing a probabilistic fingerprint generation process. Essential is that the columns of \mathbf{X} are generated according to specific probabilities $p_i, i \in \{1, \dots, m\}$, for which the probability density function is symmetric around $1/2$ and has a high bias towards its bounds.

7.1.1 Construction

The algorithm starts calculating the accusation scores and defining the *ranking vector* and *ranking 'matrix'* in a preparation stage. This is followed by a deeper search for suspected fingerprints in the core tracing algorithm.

Preparation

Accusation score: First, the accusation scores are calculated according to equation (6) in chapter 4.5.

Ranking vector: Each position of $y = (y_i)_{i=1, \dots, m} = (y_1, \dots, y_m)$ is considered according to the corresponding probability vector $p = (p_i)_{i=1, \dots, m} = (p_1, \dots, p_m)$.

As described in chapter 4.3, p_i stands for the probability of the symbol '1' in column i of \mathbf{X} . For the counter probability, the probability for a '0' in column i , we write $1 - p_i$. The probability of the symbol y_i is denoted as p_{y_i} . The lower p_{y_i} , the more significant is its amount for the accusation score, see chapter 4.5. This is used to create a significance ranking for y . We compute every value

$$k(i) := |1 - (p_i + y_i)|, \quad i \in \{1, \dots, m\}. \quad (9)$$

Hence, the closer $k(i)$ is to 0, the higher is its ranking. For example, in case $y_i = 1 \approx p_i$, $k(i)$ will be close to 1. The same holds for $y_i = 0 \approx p_i$. Other constellations will lower the (positive) value for $k(i)$.

A *ranking vector* $V = V_1, \dots, V_m$ is filled with the indices of the positions of y from highest ranking to lowest. This means, for the first entry of V it holds $V_1 = \bar{i}$ where $k(\bar{i}) = \operatorname{argmin}\{k(i)\}$. For each column of \mathbf{X} there is one entry in V .

Ranking 'matrix': We consider the k columns of \mathbf{X} for which the corresponding positions of y achieve the highest rankings: $X(:, V_1), X(:, V_2), X(:, V_3), \dots, X(:, V_k)$. Within each of those

selected columns, we are interested in all entries providing the same symbol as y in the corresponding position, i.e. in column i we search the entries for which it holds $y_i = X(j_1, V_i) = X(j_2, V_i) = \dots = X(j_l, V_i)$. The corresponding indices j_1, \dots, j_l of those entries are sorted into a structure denoted as *ranking matrix* R . Hence, the first row of R is filled with the indices of column $V_1 = \bar{i}$ of the fingerprint matrix \mathbf{X} providing the same symbol as $y_{\bar{i}}$. The second row of R is filled with the indexes of column V_2 having the same symbol as y in position V_2 , and so on.

$$R(r) = \{j_1, \dots, j_l | X(j_1, V_r) = X(j_2, V_r) = \dots = X(j_l, V_r) = y_i\},$$

for every row r of R .

Core tracing algorithm

The core tracing algorithm is divided into several steps. The first two steps directly counter the minority vote attack and the majority vote attack respectively. In many cases the algorithm outputs one or more colluder-fingerprints already after these two steps. Otherwise or if it is desired to catch more colluder-fingerprints, the core tracing algorithm can be expanded (with arbitrary complexity). The core tracing algorithm intends to reconstruct the manipulated fingerprint y , but it can be stopped prior to that as its abortion criteria can be adjusted to the individual need. Proofs and reasons of the following statements are explained below in subsection 7.1.2.

- 1: **Single entries of R :** If there is a row j_R in R that contains only a single entry, the corresponding fingerprint $X_{R(j_R,1)}$ is a colluder-fingerprint and is output by the tracing algorithm. $\mathbf{T} = \{X_{R(j_R,1)}\}$
- 2: **Indexes in each row of R :** Count the number of appearances of every index within R . If the index that appears most at the same time belongs to the fingerprint with the largest accusation score, this fingerprint is output as colluder-fingerprint.
- 3: **Reconstruct the collusion:** The fingerprints associated to the indexes appearing most in R , especially the indexes in the first row, are taken to reconstruct y .
- 4: **Successive search through R :** If one or more colluder-fingerprints have been exposed, create a vector y' in the following way. Define y' as $y' := y$. Discard every entry in y' , in case the corresponding symbol equals the symbol(s) in position i' of one or more of the exposed colluder-fingerprints. Hence y' exists of positions the already exposed colluder-fingerprints could not have created. Create a new set of rows R' in the following way. Execute the steps *Accusation score*, *Ranking vector* and *Ranking 'matrix'* R once again utilizing y' instead of y . The resulting ranking matrix is denoted as R' . Execute step *Reconstruct the collusion* with the new results of R' in order to reconstruct y' . In case this is possible, the corresponding entries in R' show up to further colluder-fingerprints. This way can be proceed until y is reconstructed completely, or until enough colluder-fingerprints are found.
- 5: **Filtering suspects:** In case the afore steps did not suffice or in case the intention was to solely apply this algorithm as a filter, the entries of R can be taken as input for a proper tracing algorithm. Thereby the input reduces from the total set of fingerprints in the fingerprint matrix \mathbf{X} to an appropriate small set of fingerprints.

7.1.2 Explanation of the algorithm

To reason the steps of the tracing algorithm above, this section contains some explanations and an example.

Preparation: Figure 9 illustrates the creation of the ranking vector V with the example of an 8 bit fingerprint. The probability vector p provides a probability of $p_4 = 0.01$ for a '1' in position $i = 4$. In the same position the attacked fingerprint $y = y_1, \dots, y_8$ contains a $y_4 = 1$. By equation (9), $k(4)$ is 0.01 as well. Because this value of k is closest to zero, the 4th column of the fingerprint matrix X is ranked highest and set as first position of ranking vector V .

Following the example of figure 9, we focus on the 4th column of the 7×8 fingerprint matrix X in figure 10. The symbols in the third and fifth row, in fingerprints 3 and 5 respectively, equal the corresponding symbol in the attacked fingerprint y of figure 9. Hence its corresponding row indexes 3 and 5 are the entries in the first row of the ranking matrix R . Which index is sorted in first is depending on its accusation scores. The fingerprint with the larger accusation score is more suspicious and for this reason favored.

Note that R is not an ordinary matrix at all, it is a list of vectors or rows containing indexes of the corresponding rows of X . Each row may contain a different number of entries.

The number of rows of R , i.e. the number of positions of V that are considered, depends on the demands of the actual distributor and hence on the code length, the number of fingerprints and on the error probabilities. It also depends on the attack strategy the colluders have chosen.

Core tracing algorithm: For a small number of fingerprints contained in the fingerprint matrix X the first rows of R frequently consist of solely one entry each. This is because of the bias generation of the fingerprints, see chapter 4.5. The closer the probability value p_i is to zero or one, the higher the probability that the entries of the corresponding column i of X all have the same symbol but one. Obviously, the less entries a column provides, the higher the probability of this event. In such an event, i.e. if there is a column in X consisting of e.g. one '0' and all other entries contain '1's, and the corresponding position in y carries a '0' as well, the corresponding fingerprint must be a colluder-fingerprint⁴. These fingerprints can directly be output by the tracing algorithm with a zero false positive error rate.

This is because the probability density function is required to have a high bias towards 0 and 1, such as in equation (5). The probability for a symbol different than the others in one specific position is very low, but on the other hand, the probability that this happens at least one times throughout the whole length of the fingerprint is comparably high.

The larger the collusion, the more fingerprints can lead to such an event, i.e. the higher is this probability. This means, if the colluders chose the minority vote attack strategy for the manipulated copy with attacked fingerprint y , there might occur a symbol that directly and undoubtedly leads to one of the colluder-fingerprints. This frequently happens also in case of a random attack strategy.

Remember that in a real watermarking application, the colluders are not able to simply *choose* a symbol, they only compare between different values of frequency bands, pixel values, etc., and choose corresponding values of frequency bands, pixel values, etc., to create the forgery. The resulting manipulated fingerprint y is not categorical altered in all detectable positions. This

⁴ Given a *perfect* watermarking algorithm underneath

p=	0.9	0.8	0.2	0.01	0.3	0.6	0.99	0.9
y=	1	0	0	1	0	1	1	0
k=	0.9	0.2	0.8	0.01	0.7	0.6	0.99	0.1
V=	4	8	2	6	5	3	1	7

V(1)=4: Column 4 has highest ranking
-> is considered first

		X(j,i)							
		0
		0
j=3		1
		0
j=5		1
		0
		0

R=

5	3
..
..
..

-> R(1)=(5,3)
-> if fingerprint 5 has a greater accusation score than fingerprint 3
-> it is more suspicious

Figure 9: Example for creation of the ranking vector V **Figure 10:** Example for creation of the ranking matrix R

fact enhances the probability of a direct identification of colluder-fingerprints after the first step 1: *Single entries of R*, consequently this example actually mirrors a realistic scenario.

On the other hand, in case the colluders followed an attack strategy alike the majority vote attack, a direct zero false positive output after the first step is not possible. This is because the majority vote decision by construction prevents from such events described before. However, for the same reason, i.e. because the majority of colluders contribute to y in every position, the first rows of $R = R(1), R(2), \dots$ are likely to include all colluder-fingerprints. According to step 2: *Indexes in each row of R*, if the fingerprint that appears most in R also gets the highest accusation score, this fingerprint can be output for participating in the collusion with a very high probability. Note again that because of the small distance to the attacked fingerprint y , as well as the large value for the accusation score that an attack strategy alike the majority vote attack brings about, we assume that most attackers presumably prevent from this strategy.

Due to what was carried out in the afore paragraphs, the first and second steps are capable to output at least one colluder-fingerprint with a high probability. The next steps concern the cases in which tracing of one colluder (or a few) is not sufficient, or for the cases in which up to this point no clear identification of a colluder-fingerprint has been possible. A closer look at R discloses the following. It is granted that at least one index of a colluder fingerprint appears in each row of R . Hence, for a majority vote attack strategy the first rows of R are likely capable of including all indexes of the colluder fingerprints. Consequently, following step 3: *Reconstruct the collusion*, a rebuilding of y with the corresponding fingerprints is possible even for larger collusion sizes.

Optionally, a deeper successive search through R can be applied by step 4: *Successive search through R*. A closer look on R reveals options to further reduce the number of fingerprints to be considered for following tracing algorithms. For the case where one or more colluder-fingerprints have been found during the first step(s), we compare y to the exposed colluder-fingerprints and create the new vector y' by discarding all positions where the exposed fingerprints and y equal. This new vector now declares which rows of R may be discarded (these are the ones generated according to a position that is no longer appearing in y') to generate R' . In addition y' also is the new vector to which we compute new accusation scores. Thereby we change the prior accusation scores by subtracting by accident high amounts that were associated to fingerprints that did not partaken in the collusion. All in all we now only consider

Table 8: False positives (FP) and false negative (FN) results for $c_0 = 3$ after step *Single entries of R* for minority vote attack and random vote attack

code length	attack strategy	FP	FN ($\mathbf{T} = \{\emptyset\}$)	$\mathbf{T} = \{c_1\}$	$\mathbf{T} = \{c_1, c_2\}$	$\mathbf{T} = \{c_1, c_2, c_3\}$
135	minority vote	0	20.5 %	43.4 %	29.5 %	6.4 %
	random vote	0	45.6 %	41.2 %	12 %	1.1 %
100	minority vote	0	26 %	44.8 %	24.7 %	4.3 %
	random vote	0	51.5 %	38.4 %	9.2 %	0.7 %

the positions of the attacked fingerprint y that the already found are not responsible for. The size of R is further reduced, the number of fingerprints to be considered is reduced respectively, and the ones within R' provide accusation scores that likely lead to the remaining the colluder-fingerprints.

These steps whether constrict the number of fingerprints to be considered or in many times already disclosed one or more of the colluder-fingerprints. If rebuilding of y is still too costly, it is possible to add another – usually cost intensive – tracing algorithm here. The effort for this algorithm should be reduced enormously by now, because due to the already filtered suspects, it has to consider decisive fewer fingerprints.

7.1.3 Evaluation

The proposed approach was tested with different parameter sets and collusion strategies. The variety of possible parameter selections and step handling of the algorithm allows a magnitude of different results. We constrict the evaluation here to two parameter sets and to step 1 and step 2 of the core tracing algorithm as it already shows competitive results and because for the subsequent steps there are no similar approaches to compare with.

In order to be comparable to the approach presented in chapter 6.2 and to the state of the art approach of [82], the first test set empirically tested the approach against a collusion of three applying the minority vote attack and the random vote attack. Accordingly, the number of fingerprints was set to $n = 100$ and the fingerprint length was set to $m = 135$ and $m = 100$ respectively. The results after step 1: *Single entries of R* are depicted in table 8. We split the results into the cases for which no colluder-fingerprint was found (false negatives; $\mathbf{T} = \{\emptyset\}$), for which exactly one colluder-fingerprint was found ($\mathbf{T} = \{c_1\}$), for which exactly two colluder-fingerprints were found ($\mathbf{T} = \{c_1, c_2\}$) and for which all three colluder-fingerprints were found ($\mathbf{T} = \{c_1, c_2, c_3\}$) already after step 1 of the core algorithm. The corresponding percentages are listed in the associated columns. As false positives did not occur, summing up the percentages of one test set yields 100%.

Since up to this point we only accuse the ones that appear in a row of their own in the Ranking Matrix R , the probability of outputting someone innocent (false positives) obviously is zero. In most cases we can already stop the algorithm at this point and provide distinguished results.

Table 8 shows, that the approach already provides acceptable results against a collusion of three fingerprints for a code length of 100 bits. This is shorter than all published 3-secure codes before including our approach for a 3-secure fingerprinting scheme presented in chapter 6.2. However, at this step of the approach, in case of a majority vote attack strategy or any alike, step 1: *Single entries of R* will not lead to any practical results at all. This is dedicated to the step 2: *Indexes in each row of R* and all that follow.

Table 9: False positive (FP) and false negative (FN) results for $m = 1.000$ and $c_0 = 6$ after the first step *Single entries of R* (minority vote, random attack) and second step *Indexes in each row of R* (majority vote attack)

attack strategy	FP	FN	$T = \{c_1\}$	$T = \{c_1, c_2\}$	$T = \{c_1, c_2, c_3\}$	$T = \{c_1, \dots, c_4/c_5\}$
minority vote	0	36.3 %	40.16 %	18.58 %	4.38 %	< 1%
random vote	0	60.57 %	31.63 %	6.86 %	< 1 %	< 1%
majority vote	0	45.92 %	54.08 %	not considered		

A false negative error rate of more than 0.5 seems insufficient, but in case the algorithm outputs a false negative error, i.e. no fingerprint is output at all, the next steps of the algorithm will highly probably identify colluder-fingerprints.

The second test set expands the collusion size to $c_0 = 6$ at a code length of $m = 1.000$ and for a set of $n = 1.000$ fingerprints. Table 9 shows the corresponding intermediate results after the first step, regarding minority vote and random vote attack, and after the second step regarding the majority vote attack strategy. Here only those fingerprints were output whose accusation score outreached the other scores by a significant amount.

In table 9 the last entries for the majority vote attack are marked as *not considered*. This is because in this (second) step we only output the one fingerprint with the highest accusation score, which is already listed in column $T = \{c_1\}$.

The approach provides shortest code lengths at satisfactory error probabilities already at this point of the tracing algorithm. If at this point some colluder-fingerprints are exposed, the applicant can choose to whether stop the tracing now or go on with the other steps to probably find further colluder fingerprints. The proposed code length in this example already reaches 1.000 bits, however existing 6-secure codes (c-secure codes at $c_0 = 6$) in literature provide code lengths that exceed our proposal by far, e.g. [84], [114], [23].

For larger collusions most results improve (inter alia) due to higher probabilities for significant positions appearing in R . An exception forms the majority vote attack strategy. Here the results for larger collusions turn worse. To balance this, an increment of the code length is inevitable.

For smaller number of fingerprints listed in the system, e.g. for parameters alike table 8 and 9, this approach acts as a tracing algorithm standing on its own. It provides practical results for the interaction of the struggling parameters code length, error probabilities and effort of the tracing algorithm.

For larger number of fingerprints listed in the system and for realistic code lengths, the first step seldom leads to a direct exposure of a colluder-fingerprint. In this case it provides a filtering of the most suspicious fingerprints and reduces the numbers of fingerprints significantly. This implies a significant reduction of the effort for a subsequent tracing algorithm that otherwise required too much effort. This is important especially for large distribution runs.

7.2 Optimized fingerprint generation for the continuous distribution

In chapter 4 we described that the fingerprinting specific processes in a fingerprinting scheme are represented by the fingerprint generation process and the tracing algorithm. In this section we present a modified fingerprint generation process and its accordingly adjusted tracing algorithm. The results are taken from our publication [11]. The idea of the approach is especially important for the approaches presented in section 7.3 and 7.4 as well as the approaches presented in chapter 8 as it presents some new ideas to the *Tardos Codes* that among others form the basis of the other approaches to come.

For probabilistic fingerprinting codes we can show that if an innocent-fingerprint was output by the tracing algorithm, the constellation of the colluder-fingerprints was very disadvantageous. In this section we propose a fingerprint generation that tries to decrease the impact of those disadvantageous constellations. This leads to a significant reduction of the error rates of the adjusted tracing algorithm.

7.2.1 Preliminaries

From chapter 4 we know that in the fingerprint generation according to the *Tardos Codes* [114], all columns of the fingerprint matrix \mathbf{X} are generated independently from each other. The entries in each column are filled randomly according to the probabilities p_1, \dots, p_m . For a very large probability of the symbol α_i in column i , only few entries in this column have the opposite symbol $|1-\alpha_i|$. This means only few rows of the fingerprint matrix \mathbf{X} will have the symbol $|1-\alpha_i|$ in position i . According to section 7.1 we insert a ranking into the probabilities. Contrary to 7.1 the manipulated fingerprint y is not considered here, as it is unknown during the fingerprint generation process.

A *significant position* represents a position in column i , for which the probability for the symbol α_i is very low. For example, if $p_i = 0.9$, i.e. a high probability for the symbol '1' in column i , all positions in column i that contain a '0' are *significant positions* as their corresponding probability $(1 - p_i)$ is very low.

A fingerprint with a number of significant positions well over the average is potentially prone to get falsely output by the tracing algorithm. This is because in a disadvantageous constellation of colluder-fingerprints, the corresponding accusation score may sum up many large positive amounts. The result is a score that exceeds the threshold Z separating the innocent-fingerprints from the suspected ones. The same holds for the other case. A fingerprint that has no significant position at all, may achieve a large accusation score as well. This is because there will be disadvantageous constellations of fingerprints that for the appropriate attack strategy will enforce positive amounts to the score at the majority of positions. Hence, to prevent the impact of disadvantageous constellations of colluder-fingerprints, as many fingerprints as possible must have a very average amount of significant positions. We achieve this by decreasing the number of potentially large innocent-fingerprints' accusation scores by means of reducing the variance of the score distribution. Thereby it is less likely that an innocent-fingerprint's score exceeds the threshold. Hence the false positive error rate can be reduced. More concerns and explanations can be found in the appendix E

7.2.2 Construction

In this approach, the columns of \mathbf{X} are no longer independent from each other, more precisely the fingerprint bits are no longer independent from the previous ones. Thereby we are able to restrain the number of significant positions.

Generation of a provisional fingerprint matrix

On the one hand, the generation of the column probabilities $(p_i)_{i=1,\dots,m}$ is adopted from [114], which means, the probabilities still follow the probability density function quoted in equation (5) of chapter 4.5. On the other hand a second vector $\nu = \nu_1, \dots, \nu_n$ is introduced. Each ν_j is allocated to row X_j of the fingerprint matrix to be generated. Since the fingerprint matrix ought to be column dependent, the columns are generated one by one, starting with the first column. The first column is filled according to the probability p_1 . The counter vector ν notices the r rows j'_1, \dots, j'_r that were assigned with a significant position. For the second column the probability p_2 dictates the distribution of the second column as ever, but in cooperation with ν . That is, in order to prevent a second significant position assigned to the same rows j'_1, \dots, j'_r in the second column again, according to the counter vector ν these rows are blocked. Hence, ν decides for each column to be generated in which rows it is forbidden to assign significant positions. After the generation of the second column, ν again notices the r' rows that now were assigned to significant positions. For the generation of the third column, the $r + r'$ rows noticed by ν (according to the information of the first and second column) are blocked for further significant positions. The probability p_3 dictates the entries accordingly. In this way the generation proceeds until the matrix is completely filled, or each row is assigned to (at least) one significant position. For the latter case, ν simply restarts again blocking significant positions at according rows as in the beginning, and the column generation continues accordingly. Thereby the matrix is filled column correlatively and the number of disadvantageously generated fingerprints is bounded. An example with a very small matrix can be found in the appendix I. Note that the generation is according to the approach presented in chapter 8.2, however, because of its small size, the example works analogously and can be taken for a correlated matrix generation as presented in this section as well.

The fingerprint matrix generated this way results in a slightly deviating distribution of the accusation scores. In [114] the variance value for the accusation scores was set to $\sigma^2 = 1$. The matrix generated as above leads to decreased variance values. Figure 11 shows the difference of the variances between the original matrix generation and our approach introducing correlations to the columns.

Generation of the fingerprint matrix \mathbf{X}

As error rates and code length strongly depend on each other, a reduction of the false positive error rate allows a reduction of the code length instead.

To achieve a reduction of the code length, we proceed as follows. The average variance value of the accusation scores is computed for many runs and for the different attack strategies listed in chapter 4.6. The average variance values for each attack strategy are compared and the average variance that achieves the largest value $\tilde{\sigma}^2$ is taken for a new adjusted formula for

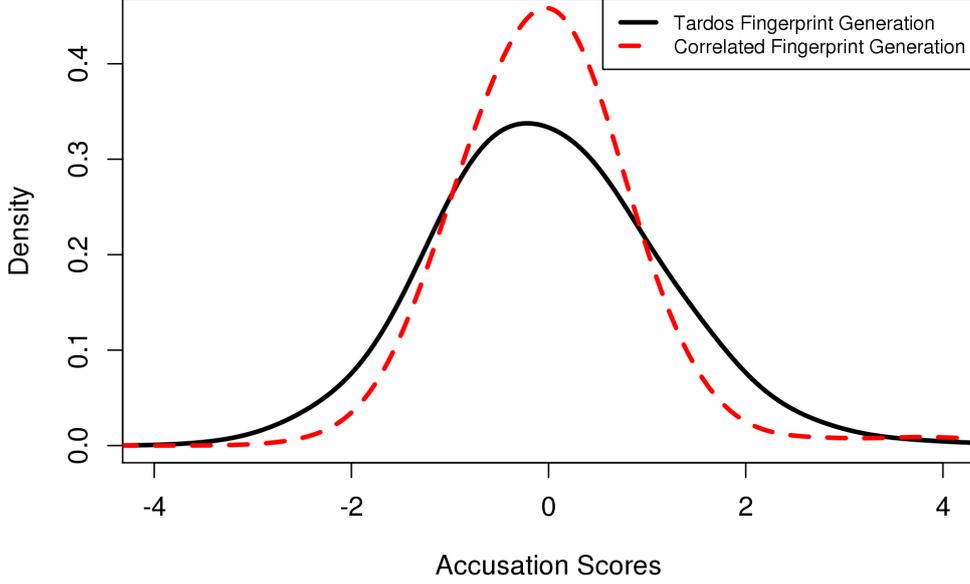


Figure 11: Histogramm of the accusation sums at *Tardos'* and our matrix generation

the code length. More precisely, we multiply $\tilde{\sigma}^2$ to the old bound on the code length m of the Gaussian assumption formula in [121] and get the new one:

$$\tilde{m} := \tilde{\sigma}^2 \cdot m = \lceil d_m c_0^2 \tilde{\sigma}^2 \ln(\varepsilon_1^{-1}) \rceil \quad (10)$$

The parameter d_m is equal to the code length parameter $d_m = \frac{\pi^2}{2}$ in [121]. We re-generate the fingerprint matrix \mathbf{X} according to the symmetric *Tardos* scheme described in chapter 4.5 applying equation (10) as code length formula. This way \mathbf{X} is generated with a significantly reduced code length at the same expected error rate.

Adjusted threshold and tracing algorithm

Because of the reduction of the code length, the threshold Z needs to be adjusted as well.

Therefore, the actual variance σ_{actual} of the accusation scores need to be calculated. The adjusted threshold is computed as

$$\tilde{Z} := \sigma_{actual} \cdot Z = d_z c_0 \ln(\varepsilon_1^{-1}) \sigma_{actual}.$$

The threshold parameter d_z is set to π according to [121]. The calculation of the accusation scores is equal to the symmetric *Tardos* scheme in equation (6) of chapter 4.5.

All fingerprints whose accusation scores exceed the new threshold \tilde{Z} are output by the tracing algorithm.

The idea to take the actual variance for the threshold calculation will be utilized again especially in section 7.4 and chapter 8.3.

7.2.3 Evaluation

The decreased variance of the distribution of accusation scores induce that it is less likely that an innocent-fingerprint's accusation score exceeds the threshold, hence the false positive error rate is reduced. We will get back to that especially in section 7.3 and in chapter 8.2. The empirical test confirm our results. We observed a slight increment of the false negative error rate compared to [121], yet it stays below 0.5. The false positive error rate, that is assumed to be more important, slightly decreases. In addition the code length was shortened by the factor of the variance $\tilde{\sigma}^2$. For example, at a maximum collusion size of $c_0 = 12$, \tilde{m} reduces to 3671, compared to 4908 in [121]. For $c_0 = 18$ we are still much shorter, $\tilde{m} = 8852$ compared to $m = 11044$ in [121].

On the downside is that the false negative error rate sometimes exceeded the selected upper bound of 0.5 for the mutual information attack in [41], see chapter 4.6. Note that this also happens for the original scheme of [121].

The computations of the variance values of the accusation sums have to be computed only once for each eligible parameter and attack strategy and can be done in advance. For this reason the effort and computational time of the whole generation process for a distributor only increases negligible in comparison to [114] and its derivatives.

The approach provides two ideas that can be applied in probabilistic fingerprinting schemes in general and which we will recall for the remaining approaches in this thesis. The one idea is to insert correlation into the columns of the fingerprint matrix \mathbf{X} in order to reduce the number of disadvantageously generated fingerprints. The other idea is to use the variance of the actual accusation scores to calculate a higher accurate threshold that is adjusted to the actual case.

7.3 Adapted fingerprint generation for discrete bias distributions

This section proposes an approach for a discrete bias distribution for generating the fingerprint matrix \mathbf{X} .

In chapter 4.3 we mentioned that apart of the continuous generation of fingerprints, i.e. probabilities following a continuous distribution function alike the *Tardos* code [114] presented in chapter 4.5, there is also the possibility of selecting a discrete distribution, e.g. [43], [84]. The prescription of generating the fingerprints accordingly was also given in chapter 4.5. Especially for maximum collusion sizes smaller than 30, some discrete distributions provide shorter code lengths than continuous distributions [85]. The approach presented in this section is a modification of the approach recently published by *Laarhoven et al.* [65] for generating the supporting points for the discrete distribution as described in chapter 4.5, which itself was an approximation to the discrete distribution construction introduced by *Nuida et al.* [84].

Our modification takes on the idea of section 7.2 introducing correlations to the column generation of the fingerprint matrix \mathbf{X} . The approach is based on our elaboration in [88]. The effect is only marginal compared to [65], but in combination with the tracing algorithm presented in the subsequent section 7.4, which introduces an adapted threshold calculation, the approach performs highly well.

7.3.1 Construction

The fingerprint matrix \mathbf{X} is generated according to [65] via the following two steps as described in chapter 4.5.

1. **Initialization:** Let $m = d_m c_0 \ln(1/\varepsilon_1)$ denote the code length and set $T = \lceil c_0/2 \rceil$. First select m bias values p_i , for $i = 1, \dots, m$, by drawing k for each value uniformly at random from $\{1, \dots, T\}$ and setting $p_i = \sin^2\left(\frac{4k-1}{8T+4}\pi\right)$.
2. **Fingerprint generation:** Each entry $X_{j,i}$ of the fingerprint matrix is selected independently from the binary alphabet with $\Pr[X_{j,i} = 1] = p_i$.

Due to the considerations in section 7.2.1, where we pointed out that the generation of disadvantageously generated fingerprints result in higher error rates, we propose to add three other steps [88]. The first step is to avoid the centricity of the bias values, i.e. to avoid that the bias values in the entries of the probability vector p are by chance centered close to 0.5. This is achieved by enforcing an average distribution of the bias values, i.e. the number of columns to be filled with probability p is proportional to the overall number of columns, see step 1 below. The second step ensures that the sum of each column actually is np_i and not deviated due to stochastic impreciseness. Note that these two regulations only matter for finite values of c_0 , as they simply enforce *average* values and prevent outliers. In the asymptotics (very large collusions) these steps will have no effect at all, for which the corresponding proofs do not differ at all from the proofs of the original versions. As a third step we ensure that each row of \mathbf{X} , aka each fingerprint, contains the expected number of '1's. This goes back to the ideas proposed in section 7.2.1, to integrate correlations to the columns of the fingerprint matrix \mathbf{X} . Here we describe an adjusted version for discrete distributions. For the same reason as in section 7.2.1 it holds, that fingerprints close to the average less frequently cause errors in the accusation. Hence reducing the number of outliers – those fingerprints that are very unlike the average – compared to the original fingerprint generation due to [65], will reduce the error probability.

The three steps are depicted in the following pseudo-algorithm. For a more detailed discussion on these steps as well as on the fingerprint generation due to [65] see the appendix F.

Define $\mathcal{S}_p = \{i | 1 \leq i \leq m, p_i = p\}$ to be the index set of the columns with bias value p . In other words, the set \mathcal{S}_p contains the indexes of all columns of the fingerprint matrix \mathbf{X} for which the probabilities p_i have the same value p .

1. Initialize the bias vector $\mathbf{p} = (p_1, \dots, p_m)$ randomly such that each probability value p occurs with the same frequency: $\lfloor m/T \rfloor \leq |\mathbf{p}| \leq \lceil m/T \rceil$ (*first optimization step*).
2. For each column i , compute the number of 1s $l_i := \lfloor np_i \rfloor$. Increment l_i by 1 with probability $np_i - \lfloor np_i \rfloor$ (*second optimization step*).
3. Iterate randomly through all positions $X_{j,i}$. For each position select $s \in \{0, 1\}$ uniformly at random.
 - a) If $s = 1$:
 - If the number of '1's in column i is smaller than l_i (*second optimization step*), and there are less than $\lceil p_i \cdot |\mathcal{S}_{p_i}| \rceil$ '1's in the sub-fingerprint $X_{j, \mathcal{S}_{p_i}}$ (*third optimization step*), then set $X_{j,i} = 1$.
 - Else, if the number of '0's in column i is smaller than $n - l_i$, and there are less than $\lceil (1 - p_i) \cdot |\mathcal{S}_{p_i}| \rceil$ '0's in the sub-fingerprint $X_{j, \mathcal{S}_{p_i}}$, then set $X_{j,i} = 0$.
 - Else, go on with 3. c)
 - b) If $s = 0$:
 - If the number of '0's in column i is smaller than $n - l_i$, and there are less than $\lceil (1 - p_i) \cdot |\mathcal{S}_{p_i}| \rceil$ '0's in the sub-fingerprint $X_{j, \mathcal{S}_{p_i}}$, then set $X_{j,i} = 0$.
 - Else, if the number of '1's in column i smaller than l_i , and there are less than $\lceil p_i \cdot |\mathcal{S}_{p_i}| \rceil$ '1's in the sub-fingerprint $X_{j, \mathcal{S}_{p_i}}$, then set $X_{j,i} = 1$.
 - Else, go on with 3. c)
 - c) If $X_{j,i}$ is not yet initiated, i.e. none of the above conditions applied:
 - if there are more '1's left for column i than '0's, i.e. if $l_i > n - l_i$, then set $X_{j,i} = 1$.
 - otherwise, set $X_{j,i} = 0$.

The first step affects the initialization step of the fingerprint generation, while the second affects the columns of the fingerprint matrix \mathbf{X} and the third step affects the rows of the matrix. Finding a matrix that fulfills simultaneously the condition from the second step as well as the condition from the third step is not trivial.

To avoid backtracking as well as to avoid that the resulting fingerprint matrix becomes too predictable, our proposed generation method will allow violating the third condition if it notices that the so-far generated matrix cannot fulfill the second and third condition simultaneously.

As discussed above, the condition in the third optimization step might be violated through c). But by our observations, this *escalation* step is only required to fill in few final positions of the fingerprint matrix \mathbf{X} .

7.3.2 Evaluation

The adapted fingerprint generation method for the discrete distribution is primary evaluated for the classical tracing scheme using the symmetric score function described in chapter 4.5. The empirical results mirror what we expected from theory, the reduced number of outliers lead to a reduced error rate.

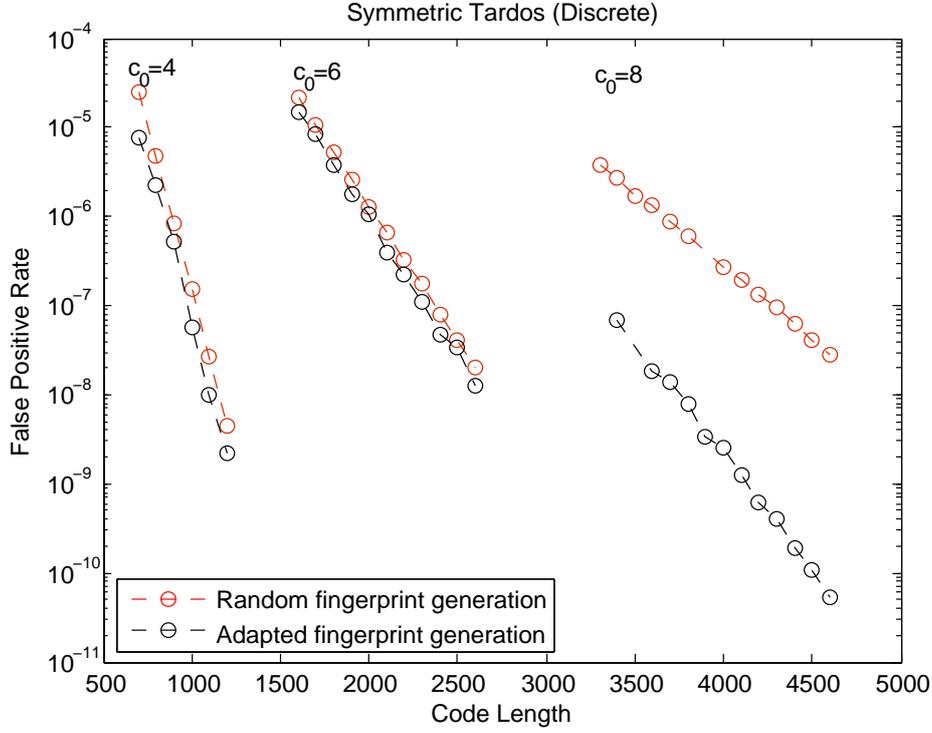


Figure 12: The false positive rates for $c_0 = 4, 6, 8$ and a fixed false negative error rate of 10^{-3} derived by the discrete bias distribution according to [65] and by our adapted approach, both combined with the symmetric score function from section 4.5, taken from [88]

Table 10: Estimated code lengths to achieve a false positive rate of about $\hat{\epsilon}_1$ for fixed false negative rate of $\hat{\epsilon}_2 = 10^{-3}$ for the discrete distribution by [65] and the symmetric score function described in section 4.5

c_0	Random fingerprint generation	Our adapted fingerprint generation
2	$2.40c_0^2(\ln(1/\hat{\epsilon}_1) + 2.76)$	$2.50c_0^2(\ln(1/\hat{\epsilon}_1) + 2.97)$
3	$3.81c_0^2(\ln(1/\hat{\epsilon}_1) + 1.85)$	$3.76c_0^2(\ln(1/\hat{\epsilon}_1) + 1.37)$
4	$3.62c_0^2(\ln(1/\hat{\epsilon}_1) + 1.55)$	$3.66c_0^2(\ln(1/\hat{\epsilon}_1) + 0.95)$
5	$4.12c_0^2(\ln(1/\hat{\epsilon}_1) + 0.62)$	$4.08c_0^2(\ln(1/\hat{\epsilon}_1) - 0.24)$
6	$3.99c_0^2(\ln(1/\hat{\epsilon}_1) + 0.38)$	$3.93c_0^2(\ln(1/\hat{\epsilon}_1) + 0.09)$
7	$4.33c_0^2(\ln(1/\hat{\epsilon}_1) - 0.18)$	$4.10c_0^2(\ln(1/\hat{\epsilon}_1) - 0.97)$
8	$4.12c_0^2(\ln(1/\hat{\epsilon}_1) + 0.10)$	$3.84c_0^2(\ln(1/\hat{\epsilon}_1) - 0.61)$
9	$4.23c_0^2(\ln(1/\hat{\epsilon}_1) - 0.02)$	$3.56c_0^2(\ln(1/\hat{\epsilon}_1) - 0.61)$

The adapted fingerprint generation method with discrete distribution functions resulted in no improvement for a chosen maximum number of colluders c_0 that was smaller than or equal to 6. For $c_0 > 6$ however, the adaption resulted in significant lower error rates of about 10-15%. This is exemplary apparent in figure 12.

Equivalent, improved error rates allow the content distributor to choose shorter code lengths while still obtaining the same security level. For the results in table 10 we empirically tested against more than 2 million attacks.

For a detailed evaluation and discussion we refer the reader to [88] and the appendix F.

7.4 Dynamic threshold computation for the interleaving score function

In this section we present a tracing algorithm that is optimized against the general requirements of a fingerprint generation process alike described in chapter 4.5 and especially alike the discrete fingerprint generation presented in section 7.3. The approach performs best against medium sized collusions as defined in chapter 5.

Other works that motivated this approach have been recently published. Recently *Oosterwijk et al.* [86] published a more general score function, denoted as interleaving score function, see equation (11) of definition 26. They proved it to be asymptotical optimal against the interleaving attack. Independently, *Huang and Moulin* [54] proved the interleaving attack to be the asymptotical optimal attack. Hence this new score function is accepted as the asymptotical, i.e. c_0 tends to infinity, optimal score function. It is left open, if this score function is also optimal in practical parameters, e.g. $c_0 < 30$. Of more importance was, that at first no method was given to calculate the accusation threshold Z . For the symmetric score function investigated in [121] described in chapter 4.5 the threshold is fixed taking the parameters chosen in advance. Adjusting this calculation for the interleaving score function is not possible, because it generates scores that heavily depend on the colluders' attack strategy and therefore the scores vary strongly. As the strategy is generally not known, a fixed threshold cannot be applied in practice.

Note that recently, *Ibrahimi et al.* [55] proposed a method to calculate a dynamic threshold especially tailored for the interleaving score function of [86]. However, it is only investigated for the asymptotic (large collusions) behavior and only provable sound to counter interleaving attacks, whereas our approach is suited for applicably small collusion sizes and independent of the attack strategy.

In this section, we revisit the idea of section 7.2 and utilize the actual accusation scores in order to propose a dynamic threshold calculation adjusted to the actual case of collusion attack and that is not depending on the collusion strategy. The results are taken from [88] and were derived independently from [55].

Our tracing algorithm requires the actual accusation scores according to [86] and the probability vector (p_1, \dots, p_m) from the fingerprint generation process, for instance from the approach in section 7.3.

7.4.1 Construction

As discussed in section 7.2, the standard deviation of the fingerprints' accusation scores is the crucial factor for the threshold calculation. In this approach, we intend to approximate the standard deviation of the (generally not known) innocent fingerprints alone, i.e. of all fingerprints in \mathbf{X} but the colluder-fingerprints. The whole scheme is as follows.

Fingerprint generation: The tracing algorithm described in the following is not depending on the fingerprint generation process. All approaches in literature for generating the fingerprints, e.g. [114], [84], [64], including the approaches explained before in this chapter, e.g. section 7.2 using a continuous distribution and section 7.3 that uses a discrete distribution, are feasible.

Tracing algorithm: Score calculation: The interleaving score function proposed by *Oosterwijk et al.* in [86] is defined as follows:

Definition 26 (Interleaving score function) For the probability p_{y_i} of the actual symbol in position i of the manipulated fingerprint y , the accusation score for fingerprint X_j is calculated as

$$S_j := \sum_{i=1}^m \varphi(y_i; X_j(i)) , \text{ with } \varphi(y_i; X_j(i)) := \delta(X_{j,i}, y_i) \frac{1}{p_{y_i}} - 1 \quad (11)$$

where $\delta(X_{j,i}, y_i)$ represents the Kronecker Delta.

The Kronecker Delta yields 1, if the symbol in position i of the manipulated fingerprint y equal the corresponding symbol in the fingerprint X_j , otherwise 0.

Tracing algorithm: Dynamic threshold calculation: The dynamic threshold is calculated as follows.

With the same fingerprinting parameters and with the same probability vector (p_1, \dots, p_m) that were used in the fingerprint generation process for filling the columns of the fingerprint matrix \mathbf{X} , another ordinary fingerprint matrix $X^{(\text{New})}$ is generated. As the fingerprints of the newly generated matrix $X^{(\text{New})}$ obviously have not partaken in any collusion, especially not in the collusion that is responsible for y , all fingerprints contained are innocent-fingerprints.

For each of the new generated fingerprints $X_k^{(\text{New})}$, with $k = 1, \dots, l$, we calculate the accusation scores $S_k^{(\text{New})}$ for the actual colluder-fingerprint y according to the interleaving score function in equation (11) of definition 26. The resulting mean and variance values are not 'falsified' by scores of colluder-fingerprints. Hence, the the corresponding standard deviation promises being a tighter approximation to the standard deviation of the scores of the actual innocent-fingerprints denoted as σ_{inn} . The corresponding sample variance for the new accusation scores $S_k^{(\text{New})}$ is computed as

$$\hat{\sigma}^2 = \frac{1}{l} \sum_{k=1}^l (S_k^{(\text{New})})^2 .$$

As the newly generated fingerprints did not participate in the forgery, $\hat{\sigma}_{\text{inn}}$ converges to σ_{inn} . This value multiplied to Z as calculated according to the corresponding chosen fingerprint generation method, represents the dynamic threshold \hat{Z} . Note that our definition of \hat{Z} does not differ from the original threshold calculation for the Tardos codes [114], because there the factor σ_{inn} was a multiplication factor in the definition of Z as well. However, there σ_{inn} was equal to 1 by construction, see chapter 4.5. For this reason the proofs can be fully adopted and the definition for \hat{Z} is sound.

Tracing algorithm: Accusation: The tracing algorithm outputs all fingerprints for which it holds $S_j > \hat{Z}$.

This way, the sample variance $\hat{\sigma}^2$ is not computed on the scores for the existing fingerprints, thus the attackers have no possibility to influence $\hat{\sigma}$. The consequence is that for an innocent-fingerprint j the ratio $S_j/\hat{\sigma}_{\text{inn}}$ is nearly identical for all examined attack strategies.

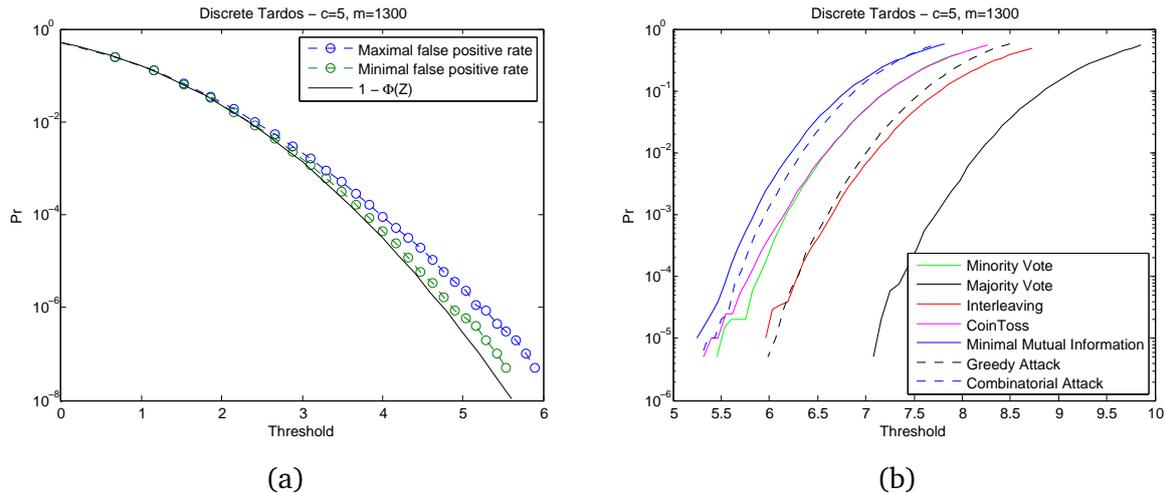


Figure 13: False positive and false negative rates applying our proposed dynamic threshold calculation for the interleaving score function and discrete bias distribution by [65]. For $c_0 = 5$ and $m = 1300$ it shows (a) the minimal and maximal observed false positive rate and (b) false negative rate for different strategies. Based on [88]

7.4.2 Evaluation

Our dynamic threshold calculation first makes the usage of the interleaving score function practically possible. In order to evaluate the approach, the value $\hat{\sigma}_{\text{inn}}$ is computed by generating 1000 new fingerprints. The false positive rate was estimated executing 20 million attacks, each time with new randomly generated colluder-fingerprints. The false negative rate was determined by averaging over 2 million attacks, each with newly generated colluder-fingerprints. For the fingerprint generation process, the discrete bias distribution by *Laarhoven* and *de Weger* [65] has been used.

The presented dynamic threshold calculation method was evaluated for collusion sizes $c_0 = 2, \dots, 9$ and each with various code lengths. It was evaluated against all attack strategies presented in chapter 4.6. The results show, that the scores of innocent-fingerprints are almost identical for all considered attack strategies, except the majority vote attack. There the scores for innocent-fingerprints become smaller, i.e. the probability of a false positive is smaller, compared to other evaluated attack strategies.

Figure 13 (a) depicts the empirically evaluated probability Pr that an innocent-fingerprint gets a score exceeding \hat{Z} for $c_0 = 5$ and $m = 1.300$.

Our results show, that the interleaving score function with the proposed method to compute $\hat{\sigma}_{\text{inn}}$ leads to lower error rates for $c_0 \geq 5$ in comparison to the symmetric score function described in chapter 4.5. For $c_0 = 3$ and $c_0 = 4$, the symmetric score function resulted in smaller error rates. For $c_0 = 2$, the interleaving score function and the symmetric score function are identical for the used discrete bias distribution.

In analogy to section 7.3, the code length was parameterized as $m = d_m c_0^2 (\xi + \kappa)$, where ξ is shorthand for $\ln(1/\hat{\epsilon}_1)$, and d_m as well as κ were determined to achieve a certain false positive rate $\hat{\epsilon}_1$ and a fixed false negative rate of either $\hat{\epsilon}_2 = 10^{-2}$ or $\hat{\epsilon}_2 = 10^{-3}$. The results of the parameterization is shown in table 11. For example, given the maximum expected number of

Table 11: Code lengths for the symmetric score function and the interleaving score function using our dynamic threshold computation and the discrete distribution by [65]. The parameter ξ denotes $\ln(1/\hat{\epsilon}_1)$. Note that for $c_0 = 2$, both score functions are identical, as $p_i = 0.5$ for all i .

c_0	Symmetric Score Function		Interleaving Score Function	
	$\epsilon_2 = 10^{-2}$	$\epsilon_2 = 10^{-3}$	$\epsilon_2 = 10^{-2}$	$\epsilon_2 = 10^{-3}$
2	$2.32c_0^2(\xi + 0.94)$	$2.40c_0^2(\xi + 2.76)$	-	-
3	$3.58c_0^2(\xi + 0.82)$	$3.81c_0^2(\xi + 1.85)$	$3.89c_0^2(\ln(1/\epsilon_1) - 2.3)$	$4.01c_0^2(\xi - 1.49)$
4	$3.41c_0^2(\xi + 0.67)$	$3.62c_0^2(\xi + 1.55)$	$4.23c_0^2(\xi - 2.48)$	$4.48c_0^2(\xi - 1.77)$
5	$3.84c_0^2(\xi - 0.06)$	$4.12c_0^2(\xi + 0.62)$	$3.29c_0^2(\xi - 3.02)$	$3.48c_0^2(\xi - 2.07)$
6	$3.77c_0^2(\xi - 0.35)$	$3.99c_0^2(\xi + 0.38)$	$3.21c_0^2(\xi - 3.30)$	$3.39c_0^2(\xi - 2.30)$
7	$4.06c_0^2(\xi - 0.80)$	$4.33c_0^2(\xi - 0.18)$	$2.48c_0^2(\xi - 0.15)$	$2.67c_0^2(\xi + 0.32)$
8	$3.97c_0^2(\xi - 0.81)$	$4.12c_0^2(\xi + 0.10)$	$2.40c_0^2(\xi + 1.16)$	$2.56c_0^2(\xi - 1.69)$
9	$4.12c_0^2(\xi - 0.99)$	$4.23c_0^2(\xi - 0.02)$	$2.12c_0^2(\xi + 1.95)$	$2.27c_0^2(\xi + 2.26)$

colluders as $c_0 = 8$, a code length of at least $m = 4.12c_0^2(\xi + 0.10)$ is necessary to observe a false positive error rate of $\hat{\epsilon}_1$ and a false negative error rate of $\hat{\epsilon}_2 = 10^{-3}$ for the symmetric score function. For the same parameters, the interleaving score function with the dynamic threshold calculation suffices a code length of $m = 2.56c_0^2(\xi - 1.69)$.

For $c_0 \geq 5$ the code length parameter d_m is much smaller applying the interleaving score function in comparison to the symmetric score function. Also, it is fairly close to the asymptotic lower bound of $d_m = 2$, that was proven by [86].

More evaluations are to be found in the appendix G.

7.5 Summary

We proposed four different approaches in order to improve the fingerprint generation and the tracing algorithm. Each approach aims at specific properties and parameters. Sections 7.1 and 7.2 generate the matrix with a continuous distribution, sections 7.3 and 7.4 make use of a discrete distribution to generate the fingerprint matrix. The approaches in section 7.1, 7.2 and 7.3 present adjusted or modified tracing algorithms for the symmetric score function in order to trace colluder fingerprints, section 7.4 also introduces a method to firstly apply the interleaving score function recently published by [86]. All approaches are focused on (but are not restricted to) medium sized collusions.

false positive error bound: All approaches result in improved empirical false positive error rates. During the tests for the ranking search approach (section 7.1) no false positive error occurred at all. The other approaches utilized the reduced false positive error rate in order to improve other parameters such as the code length, and therefore the false positive rate was fixed.

false negative error bound: The presented tests for the ranking search approach only consider the first and second step of the whole tracing scheme. Already then the selected rate of 0.5 was satisfied. For reason of comparison, the approaches in sections 7.3 and 7.4 were tested with a fixed selected choice for the false negative error rate. However, the correlated matrix approach for the continuous distribution (section 7.2) achieved slightly increased false negative rates compared to its predecessor [121].

code length: All approaches achieve significant reductions of the code lengths compared to its corresponding preceding fingerprinting scheme.

number of fingerprints: The ranking search approach is mainly restricted to scenarios with a small number of fingerprints. Depending also on the other parameters, the approach will lose its effectiveness, i.e. directly outputting colluder-fingerprints in step 1 or step 2 of the algorithm, with increasing number of fingerprints and colluders (e.g. $n = 10^4$ and $c_0 = 10$). The other approaches in this chapter are not noteworthy affected by the number of fingerprints alike all *Tardos Codes* in literature.

complexity: The complexity for the ranking search approach varies in dependency of the selected steps within the tracing algorithm. The effort is very low stopping after the first two steps, but it can increase arbitrarily during the further steps. In that case the approach is intended to serve as a filter instead, reducing the effort for adjacent tracing algorithms. The other approaches result in increased complexity compared to its corresponding predecessors. This is because each works operating additional steps in order to improve the state of the art. The additional complexity however remains negligibly.

maximum chosen collusion size: As mentioned, the ranking search approach is most effective for small values of c_0 ($c_0 \in \{3, \dots, 6\}$). The correlative fingerprint matrix generations (sections 7.2 and 7.3) lose in effectiveness for increasing values of c_0 , but achieve significant improvements for all medium sized selections. The dynamic threshold calculation (7.4) is mainly tested against $c_0 \in \{2, \dots, 9\}$ but as it is the first approach to apply the interleaving score function of [86] in practice, it is not restricted to small values of c_0 .

Two major important awarenesses are taught in this chapter. The first is that the additional conditions in the fingerprint matrix generation process are able to reduce fingerprint outliers that are disadvantageously generated. The second is that both the discrete distribution for the

fingerprint generation and the interleaving score function, but also its combination, allow to significantly reduce the code length also for practically small choices of c_0 .

8 Short and efficient fingerprinting codes

The approaches we proposed in chapters 6 and 7 are optimized against a very small to medium sized upper bound of colluders c_0 . They serve well scenarios for which a relatively small value for c_0 is required. For large values we need other fingerprinting schemes or adjusted and modified ones.

Various publications consider very large ($c_0 \rightarrow \infty$) collusions, e.g. [23], [21], [76] and others already mentioned, [114], [121], [51], [50], [52]. Our approaches presented in this chapter focus on maximum collusion sizes of c_0 from 15 to at most 30. Compared to smaller values for c_0 , such as during the last chapters, the approaches in literature achieve better results for larger collusion sizes, e.g. $c_0 = 15, 16, \dots, 30$. However, for the same reasons (impractical assumptions regarding c_0) these schemes still lack of applicability.

We present three different approaches. The first approach follows the idea of [74] proposing a joint tracing algorithm, see chapter 4.3. We propose a combination of a single tracing and a subsequent joint tracing algorithm in order to reduce the code length. The approach is based on our work in [10].

The second approach proposes an enhancement of the correlated fingerprint generation approach already proposed in chapter 7.2 based on our results of [11]. Introducing a rank correlation coefficient for the fingerprint matrix generation, we further reduce the error rates.

The third approach also goes back the approach of chapter 7.2. We propose a modified tracing algorithm applying a dynamic threshold calculation using mixture models. The idea to that approach is taken from our works in [71] and was recently published in [96]. The new threshold calculation can be applied on any score function, including the interleaving score function, see chapter 7.3. It is independent of the attack strategy and it remains reliable also in case the actual number of colluders differ from the chosen upper bound, i.e. $c_0 \neq c$.

8.1 Efficient joint decoding tracing algorithm

In this section we present a joint tracing algorithm based on the idea of *Meerwald* and *Furon* [74] that utilizes our correlative fingerprint generation process presented in chapter 7.2 ([11]). The reduction in the code length achieved by the correlative fingerprint generation process and the idea of adaptive thresholds for the tracing algorithm according to chapter 7.2 are combined with a joint decoding tracing approach that reduces the error probabilities. The approach is published in [10].

In a joint tracing algorithm, each accusation score is no longer calculated between one single fingerprint X_j and the manipulated fingerprint y according to a single accusation score as defined in equation (12) of chapter 4.3. It is calculated according to tuples of fingerprints X_{j_1}, \dots, X_{j_l} and y according to a joint accusation score as described in equation (13). The advantage of a joint tracing algorithm compared to a single tracing algorithm is a reduced error probability that can be traversed into shorter code lengths. On the downside is the significant increment of computational complexity. For compensation reasons, in our approach we combine a joint tracing algorithm with a single tracing algorithm in order to gain the advantages of both.

We apply the single tracing algorithm of [121] described in chapter 4.5 to primary reduce the number of fingerprints to be considered for the following steps of the joint decoder. Similar to the approach introduced in chapter 6.2, the resulting fingerprints are used to extract the most likely attack strategy. The resulting information is taken as input for the maximum likelihood estimator (MLE) applied in a modified joint decoding approach according to [74] to output the most probable colluder-fingerprints.

Compared to the approach of [74], our approach also significantly reduces the complexity of the tracing algorithm. This allows to extend our approach also to larger collusion sizes, for which other joint decoding approaches demand impractically high efforts.

8.1.1 Preliminaries

The fingerprint generation process is taken from chapter 7.2. The corresponding formula for the code length is

$$m = \sigma^2 \cdot m_{\text{škorić}} = \frac{(c_0 \pi \sigma)^2}{2} \ln \frac{1}{\varepsilon_1 \sqrt{2\pi}}.$$

The tracing algorithm uses the following notations and definitions.

- X_{joint} denotes the subset of fingerprints that are still under supervision. Its number $|X_{\text{joint}}|$ decreases with every step of the algorithm.
- The actual step of the joint decoding algorithm is represented by parameter T . Note that this equals the actual subset size.
- With X_{SI} we denote the actual $T - 1$ subset of fingerprint indexes of those fingerprints that are determined already as colluder-fingerprints and that will be output by the tracing algorithm. The notation bears on the side information that it adds to the corresponding tracing step.

- The subset $\tau_T(j)$ consists of the fingerprint X_j currently considered and of subset X_{SI} : $\tau_T(j) = \{j, X_{SI}\}$.
- May K denote a subset of fingerprint indexes and X_K the corresponding matrix, then $\mathbf{b}_{y_i}(K)$ is defined as the number of occurrences of symbol y_i in column i of X_K , in position i of the manipulated fingerprint y , respectively.
- We use the *channel information* $Ch_\theta(j, j')$ as a measurement for the interaction between the actual considered fingerprints X_j and $X_{j'}$ and the manipulated fingerprint y .
- The *channel information vector* $\theta = \theta_{c_0}$ denotes the output vector of Ch_θ . It represents the assumed type of collusion strategy.

The channel information Ch_θ and its output vector θ will be considered in more detail in the subsequent section.

8.1.2 Construction

The tracing algorithm is divided into four steps of which the last step is an iterative algorithm. The first step is the adjusted tracing algorithm described in chapter 7.2, that intends to reduce the number of suspicious fingerprints. Step 2 gives an estimation of the collusion strategy and an according vector θ , it is adjusted from the approach of [74], though the values are derived through empirical examination only. With the third step, the first pair to be set as colluder-fingerprints is determined. We make use of the log-likelihood ratio to sort out the fingerprint pair having the highest probability of having pertained in the collusion. The corresponding score is an adjusted version of the score applied in the approach of [74]. It was shown that the score is optimal in the *Neyman-Pearson* sense [77] in case the collusion vector θ was known (as the collusion is never known, this is never true, for which we need the estimation in step 2). The error probabilities for the log-likelihood ratio in this step as well as in the (iterative) step 4 of the algorithm strongly depend on the number of iterative steps and the estimated collusion strategy. Because they are analogue to [74], we will not examine this here.

Step 1: Pruning fingerprints by single decoding: The accusation scores for all fingerprints X_1, \dots, X_n are computed according to [121] described in chapter 4.5.

$$S_j := \sum_{i=1}^m \delta_{y_i, X_{ji}} g_1(p_{y_i}^{(i)}) + [1 - \delta_{y_i, X_{ji}}] g_0(p_{y_i}^{(i)}).$$

Those fingerprints whose accusation scores exceed the threshold

$$\tilde{Z} := \pi c_0 \ln \left(\frac{1}{\varepsilon_1 \sqrt{2\pi}} \right) \sigma_{actual}$$

are grouped into the set X_{joint} .

Step 2: How to handle Theta: The decision about the collusion strategy the colluders probably applied is brought about by the mean of the channel information Ch_θ between all possible pairs of fingerprints $\{X_j, X_{j'}\} \in X_{joint}$, $j \neq j'$, and the manipulated fingerprint y . It is calculated as

$$Ch_\theta(j, j') = \sum_{i=1}^m \delta_{y_i, (X_{j,i}, X_{j',i})} (1 - p_{y_i})^{c_0 - 1}$$

where $\delta_{y_i, (X_{j,i}, X_{j',i})}$ equals 1 if minimum one of the fingerprints $X_{j,i}$ and $X_{j',i}$ has the same symbol in position i as the manipulated fingerprint y . The parameter p_{y_i} represents the corresponding probability for the symbol that occurs in y . Depending on the mean of the channel information \overline{Ch}_θ , we select the value for θ .

$$\theta = \begin{cases} [0, 0, 0, \dots, 1, 1, 1], & \text{if } \overline{Ch}_\theta < 5 \\ [0, 0.5, 0.5, \dots, 0.5, 0.5, 1], & \text{if } 5 \leq \overline{Ch}_\theta < 20 \\ [0, \frac{1}{c_0}, \frac{2}{c_0}, \dots, \frac{c_0-1}{c_0}, 1], & \text{if } 20 \leq \overline{Ch}_\theta \end{cases}$$

The size of the vector is $|\theta| = c_0$. In case it holds $\overline{Ch}_\theta < 5$, the vector θ is filled with '0's for the first $c_0/2$ positions, the rest of the vector is filled with '1's.

Step 3: Selecting the first two suspicious fingerprints: Compute pair-scores S_{pair} as described below between the manipulated fingerprint y and all possible pairs of fingerprints $\{X_j, X_{j'}\}$, $j \neq j'$, contained in X_{joint} .

$$S_{pair}(j, j') = \sum_{i=1}^m \frac{\mathbb{P}[\mathbf{b}_{y_i}(\{j, j'\}), 2, p_{y_i}, \theta]}{\mathbb{P}[0, 0, p_{y_i}, \theta]}$$

The probability $\mathbb{P}[\cdot]$ is computed as

$$\mathbb{P}[b, T, p, \theta] = \sum_{k=b}^{c_0-T+b} \theta(k) \binom{c_0-T}{k-b} p^{k-b} (1-p)^{c_0-T-k+b}. \quad (12)$$

A derivation for equation (12) can be found in [74]. It describes the probability for y_i in the i th position of y , knowing that the same symbol has been distributed to fingerprints with probability p_{y_i} , the collusion model θ , and the identity of T colluder-fingerprints having b symbols y_i and $T - b$ symbols $|1 - y_i|$.

Next, for each fingerprint X_j sum up the scores of all pairs $\{X_j, X_{j'}\} \in X_{joint}$ in which he participated. The one fingerprint with the lowest sum is discarded from X_{joint} and no longer considered. The fingerprint pair with highest sum is also discarded from X_{joint} , it is sorted into the group X_{SI} containing those fingerprints that are definitively output by the tracing algorithm.

Step 4: Subset size 3 and more: With X_{SI} containing the first two fingerprints, we consider the next subset size $T = 3$. The iterative core of the algorithm is started as follows.

- i For every step T of the algorithm, the subset $\tau_T(j)$ is built for each fingerprint X_j left in X_{joint} . Also the corresponding score S_{subset} is computed as

$$S_{subset}(\tau_T(j)) = \sum_{i=1}^m \frac{\mathbb{P}[\mathbf{b}_{y_i}(\tau_T(j)), T, p_{y_i}, \theta]}{\mathbb{P}[\mathbf{b}_{y_i}(X_{SI}), T-1, p_{y_i}, \theta]}, \quad (13)$$

with $\mathbb{P}[\cdot]$ according to equation (12).

- ii In every step T the fingerprint with the highest score according to equation (13) and the fingerprint with the corresponding lowest score are discarded from X_{joint} . The

Table 12: Comparison of code lengths for chosen bounds of $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 0.5$ and $n = 10.000$ fingerprints

Tracing strategy	c_0					
	10	12	14	16	18	20
Single tracing: Symmetric Tardos [121]	6365	9165	12474	16293	20620	25457
Joint tracing: Our approach	~ 2479	~ 3671	~ 5032	~ 6725	~ 8852	~ 11058

one with the highest score is sorted into X_{SI} , i.e. the group of fingerprints that will be output, the one with the lowest score is no longer considered.

- iii If the set of fingerprints to be output has reached the size of the maximum chosen collusion size $|X_{SI}| = c_0$, or if the set of fingerprints to be considered left is empty, $X_{joint} = \emptyset$, the algorithm stops and outputs X_{SI} . Otherwise the algorithm increases T by 1 and continues going back to step i.

The number of fingerprints in X_{joint} after step 1 is approximately the same as the chosen maximum number of colluders c_0 . Oftentimes all colluder-fingerprints are already separated from the innocent-fingerprints. This is because, the threshold \hat{Z} (which coincides to the shorter code lengths) causes a high percentage of the accusation scores of the colluder-fingerprints to exceed \hat{Z} , which means they stay under supervision in the further procedure of the algorithm. However, for the same reason also innocent-fingerprints are constantly appearing in X_{joint} .

Within the description of the channel information vector θ , the parameter n , representing the number of fingerprints listed in the system, does not appear. This is due to its minor dependency on *Tardos Codes* in general that was mentioned before. The only influence of n for the proposed code comes from the code length computation. We propose the condition $n \leq \varepsilon_1^{-1}$. Hence, the effect of n on the vector θ is negligible.

8.1.3 Evaluation

To evaluate our approach for a joint decoding strategy, we considered the code length m and the complexity at chosen bounds on the false positive error of $\varepsilon_1 = 10^{-3}$ and on the false negative error of $\varepsilon_2 = 0.5$.

A digest of the results regarding the reduced code lengths are listed in table 12. We compared our joint decoding approach to the symmetric Tardos code [121] described in chapter 4.5. This comparison was chosen for reasons of generality and because other joint decoding approaches so far in our mind are determined as too complex regarding the selected values for code length and maximum chosen collusion size. The resulting code lengths are reduced significantly for all presented maximum collusion sizes.

In appendix H there will be more results and a comparison to e.g. the joint decoding approaches of [74] for reduced collusion sizes.

Table 13 shows the false positive errors and the false negative errors for collusion sizes of $c_0 = 10$ and different attack strategies. The experimental results for the errors are far below their chosen bounds of $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 0.5$.

Table 13: Experimental results for different attack strategies with code length of $m = 2.479$ at $c_0 = 10$, $\varepsilon_1 = 10^{-3}$, $\varepsilon_2 = 0.5$ and $n = 10.000$ fingerprints

Error rate	Attack strategy				
	Random vote	Minority vote	Majority vote	Interleaving	Worst case
False positive	$2 \cdot 10^{-5}$	$20 \cdot 10^{-5}$	0	0	0
False negative	$188 \cdot 10^{-5}$	$164 \cdot 10^{-5}$	$164 \cdot 10^{-5}$	$264 \cdot 10^{-5}$	$1456 \cdot 10^{-5}$

Note that the chosen bound on the false positive error was set to $\varepsilon_1 = 10^{-3}$ instead of $n \leq \varepsilon_1^{-1}$. This was done for reasons of comparison and because of the reduced effort for the testing.

The attack strategy referred to as *worst case* belongs to the class of stateful attacks as described in chapter 4.6 and works similar to the *Greedy Attack* of that chapter. For its explanation please see in the appendix H.

Our approach is very effective regarding its computational complexity. Hence this approach is the first in literature to allow joint decoding tracing for comparably large values of c_0 (i.e. $c_0 \geq 12$). The complexity of the proposed code basically depends on the number of fingerprint indexes listed in X_{joint} after the single decoding step 1 according to [121]. It can be computed as the number of score calculations ϑ according to

$$\vartheta = \binom{|X_{\text{joint}}|}{2} + \sum_{k=1}^{c_0 - (2k-1)} |X_{\text{joint}}| - (2k + 1).$$

This means, for instance within $c_0 = 8$ our approach suffices $\vartheta = 37$ score calculations from step 3 onward. At the same time the approach in [74] costs $8 \cdot 10^6$ score calculations.

8.2 Optimized fingerprint generation for continuous distributions II

In chapter 7.2 we presented our idea of an optimized fingerprint matrix generation by introducing correlations to the columns of the fingerprint matrix \mathbf{X} , that was published in [11]. The idea was taken on by the approach presented in chapter 7.3 for discrete bias distributions. This section presents an enhancement of the idea proposed in chapter 7.2, this time for continuous distributions.

Recalling the observations from chapter 7.2, we know that inserting a correlation into the column generation for the fingerprint matrix \mathbf{X} leads to a decreased standard deviation of the accusation scores of innocent-fingerprints. Here we elaborate the approach and its impact on the rank correlation coefficients of \mathbf{X} by means of descriptive statistics. Finding a correlation bound, i.e. an abort criteria of the correlation, in such a way that the correlation is not verifiable – or only to a minimum extent – allows generating a fingerprint matrix \mathbf{X} with the highest probability of no disadvantageously generated fingerprints.

In the following is listed the construction and some digest of the evaluation. The statistic explanations to the approach are shifted to the appendix I.

The approach requires the fingerprinting parameters code length m , maximum expected collusion size c_0 , number of fingerprints n and selected maximum error rates of ε_1 and ε_2 .

8.2.1 Construction

The generation of the fingerprint matrix \mathbf{X} requires the selection of a continuous distribution function according to chapter 4.5.

From the corresponding distribution we create a ranked probability vector \hat{p} similar to chapter 7.1. But this time without the influence of the manipulated fingerprint y instead solely by permuting the original probability vector p . We create $\hat{p} = (\hat{p}_1, \dots, \hat{p}_m)$ so that it holds

$$|\hat{p}_i - 0.5| \leq |\hat{p}_{i+1} - 0.5| \quad \forall i \in \{1, \dots, m-1\}$$

The columns of \mathbf{X} are filled successively.

- In column i , if it holds $[\hat{p}_i \cdot n] < \frac{n}{2}$, all entries in column i are initialized with '0'. Afterwards $l := [\hat{p}_i \cdot n]$ entries in i are exchanged with '1's. If possible the entries are selected normally distributed.
- Otherwise, i.e. if holds $[\hat{p}_i \cdot n] \geq \frac{n}{2}$, the entries are initiated with '1' and the $l := n - [\hat{p}_i \cdot n]$ entries are exchanged with '0'.
- All j_1, \dots, j_l indexes of the entries that were exchanged, are locked for the next columns. This means, for the next column generation processes, it is prohibited to exchange the entries in positions j_1, \dots, j_l again.

The entries are locked until all other rows are locked as well. This way the next columns of matrix \mathbf{X} are generated until all positions are locked. In that case all locks are nullified and the process continues until the matrix is completely filled. An example for the generation of a correlated fingerprint matrix (with exemplary small dimensions) can be found in the appendix I.

Table 14: Results for the error rates (FP = average false positive error, $|C|$ = average number of colluders caught) with uncorrelated and correlated fingerprint matrix \mathbf{X}

attack strategy	c_0	λ	FP	$ C $	max. std. deviation
interleaving	21	0.0	0.622	10.435	1.06
		0.1	0.057	11.065	0.88
	31	0.0	0.576	15.744	1.08
		0.1	0.040	15.796	0.89
minority vote	21	0.0	0.677	11.833	1.07
		0.1	0.391	10.436	1.01
	31	0.0	0.622	14.802	1.07
		0.1	0.294	14.301	0.99
majority vote	21	0.0	0.639	10.423	1.06
		0.1	0.024	11.217	0.84
	31	0.0	0.618	15.438	1.08
		0.1	0.101	15.959	0.86

Note that until this point, it is only a slightly different prescription for generating the columns of \mathbf{X} compared to the approaches of chapter 7.2 and 7.3.

The correlation between the columns is evoked by these locking of column positions for the subsequent column generation steps. For each column generation step we verify if it holds $\hat{p}_i \in \mathcal{B}_\epsilon(\lambda)$, where $\mathcal{B}_\epsilon(\lambda) := \{x \in \mathbb{R} : |x - \lambda| < \epsilon\}$ denotes the ϵ -environment of the correlation bound $\lambda \in [t, 1 - t]$ and t is the cutoff parameter defined in chapter 4.5. In case $\lambda = 0.5$ the fingerprint matrix is fully correlated, in case $\lambda = t$ the matrix remains uncorrelated. Tests have shown, that the correlation of \mathbf{X} inserts negligible alterations into the distribution of the correlation coefficients until a value of $\hat{p}_i \approx \lambda = 0.10$. In section 8.2.2 we will present results for which this choice of λ leads to significant decrement of the false positive error rate (about a factor of 6) while the false negative remains the same (or decreases as well). More details regarding the correlation bound λ and the choice of the cutoff t are sorted into the appendix I.

8.2.2 Evaluation

We evaluated the correlative matrix generation in different test sets. The parameter set was chosen as $n = 1000$, $c_0 = \{21, 31\}$, $\epsilon_1 = \frac{1}{n}$, $\epsilon_2 = \frac{1}{2}$ and $t = \frac{1}{n}$. We generated the fingerprint matrix \mathbf{X} for $\lambda = 0$ (representing the standard matrix generation described in chapter 4.5) and $\lambda = 0.1$ (correlative). Both matrices were tested against 1.000 collusion attacks. We applied the symmetric Tardos (chapter 4.5) for the colluder tracing and for the evaluation of the error rates.

The results are depicted in table 14. We listed the false positive error rate and the false negative error rate and also the standard deviation of the accusation scores of innocent-fingerprints.

For all cases we observe a clear reduction of the false positive error, while the false negative rate remains comparatively stable. Analogue to the observations of [11] explained in chapter 7.2 the standard deviation of the innocent-fingerprints decreases for correlative fingerprint matrices.

From the different values in column *maximal standard deviation*, we see that the column correlation has the most effect on the majority vote and the interleaving attack. The results for the average number of false positives and the average number of output colluder-fingerprints are the logical consequence of this observation. All in all we see that for the same parameters no column correlation ($\lambda = 0.0$) compared to column correlation by $\lambda = 0.1$ result in significantly decreased error rates.

A deeper analysis is devoted to the appendix I.

8.3 Universal threshold calculation for large collusions

For the approach proposed in this section, we revisit the idea of our approach published in [11] and presented in chapter 7.2. There the goal was to decrease the standard deviation of the accusation scores of innocent-fingerprints and to calculate an adjusted threshold. The approach presented in this section is published in [96]. Preliminary work was done in [71].

In chapter 7.3 we presented an approach for discrete distributions, that decreases the standard deviation of the scores of innocent-fingerprints to be secure against a relatively small number of colluders ($c_0 < 20$). The corresponding dynamic threshold calculation was introduced in 7.4 that also allows applying the interleaving score function of [86] without the knowledge of the actual attack strategy.

However, for large values of c_0 the chosen bound on the false negative error ε_2 can no longer be guaranteed. Due to statistical impreciseness, the larger the collusion size the larger will be the deviation from this bound. In addition, the advantage for the discrete distributions vanishes for larger choices of c_0 . It was shown that in the asymptotics the rates for the discrete generation tend to their continuous counterpart [62]. What is more important for this chapter's use case, is that with discrete distributions a huge drop in the performances occurs, when c is indeed bigger than the expected c_0 . Such a thing cannot happen with continuous distribution (smooth drop as c increases) [87].

In this section, we propose an approach for a dynamic threshold calculation for larger values of c_0 that can be applied for both discrete and continuous fingerprint matrix generations. The approach can be seen as a universal method for any accusation function, as it solely works with the scores calculated in the actual case of collusion attack. It does not need any information about the collusion strategy nor its size. The adjustment of the scores is achieved by employing two-component mixture models.

8.3.1 Construction

Mixture models are linear superpositions of (two) arbitrary distributions that interfere with each other. We want one distribution to represent the scores of the innocent-fingerprints and the other distribution to represent the scores of the colluder-fingerprints.

Note that for smaller values of c_0 the scores of the colluder-fingerprints will only be detected as single outliers and not as belonging to a decent distribution, simply because they are too few. For this reason the application of mixture models first makes sense for larger values of c_0 , e.g. $c_0 \geq 15$.

A two-component mixture model is defined as

$$\phi(x|\Theta) = \alpha_1\rho_1(x|\theta_1) + \alpha_2\rho_2(x|\theta_2) \quad (14)$$

for ρ_1, ρ_2 denoting static probability density functions with parameter sets θ_1 and θ_2 and where $\alpha_1 > 0$ and $\alpha_2 > 0$ are the associated weight parameters that satisfy $\alpha_1 + \alpha_2 = 1$. The total parameter set is denoted as $\Theta := \{\alpha_1, \theta_1, \alpha_2, \theta_2\}$.

We decide for mixture models consisting of two normal distributions, referred to as *Gaussian Mixture Model* (GMM) [18]. These are adjusted to the two clusters that represent the accusation scores of innocent-fingerprints and colluder-fingerprints. The choice for a Gaussian mixture

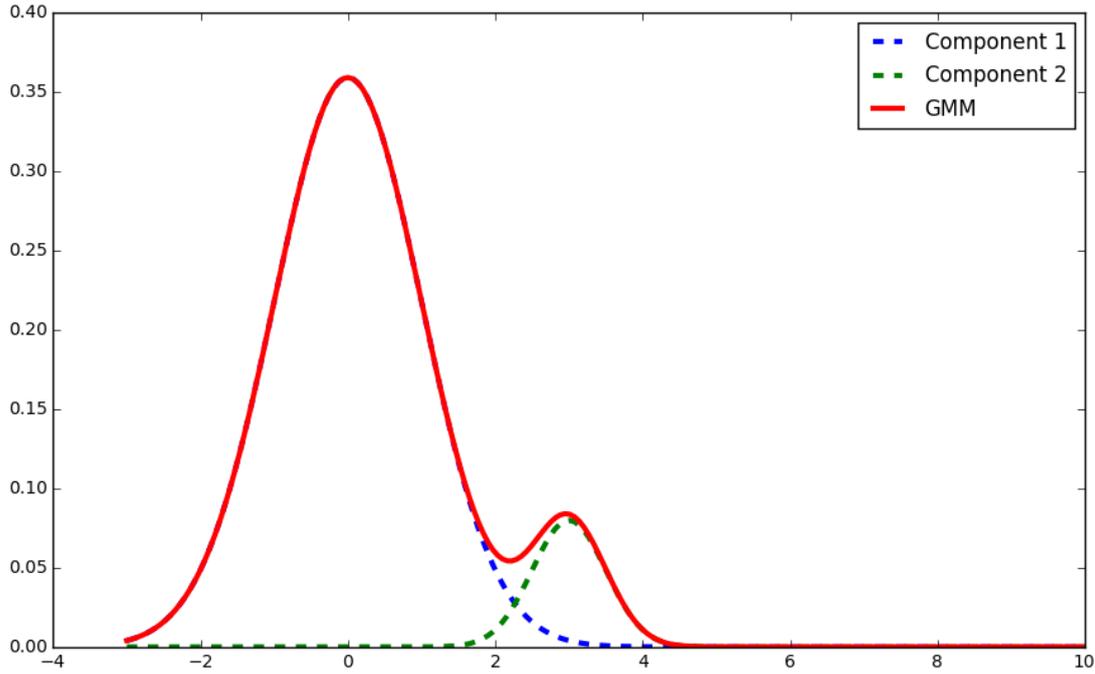


Figure 14: Example of a two-component mixture model consisting of two normal distributions.

model is assumed as sound. Already in [124] it was shown that by construction the *Tardos* codes yield curves for the scores of innocent-fingerprints and for the scores of colluder-fingerprints, that are assumed as normally distributed. Hence, the probability functions of equation (14) are normal distributions $\rho_r(x|\theta_r) = \mathcal{N}(x|\mu_r, \sigma_r^2)$ with distribution parameters $\theta_r = \{\mu_r, \sigma_r^2\}$ for $r \in \{1, 2\}$. An example of a two-component mixture model consisting of two normal distributions with $\phi(x) = 0.8\mathcal{N}(x, 0, 1) + \mathcal{N}(x, 3, 0.5)$ is depicted in figure 14.

EM for the Gaussian Mixture Model

The goal is to determine the parameter set, containing the mean and the variance for each component. This is achieved applying the iterative *Expectation Maximization* (EM) algorithm. The EM is an approved method for finding a *maximum-likelihood-estimator* of the parameters of a mixture model in case not all data is known [18], see the appendix J. Thereby we are able to find a maximum likelihood estimator for the parameters of the mixture distribution that represents the accusation scores. The construction is as follows:

1. **Initialization:** Estimate the first parameter set: $\Theta^{(0)} = \{\alpha_1, \mu_1, \sigma_1^2, \alpha_2, \mu_2, \sigma_2^2\}$.
2. **Expectation step:** In step k , calculate $\tau_{1,j}^{(k)}$ and $\tau_{2,j}^{(k)}$ for all $j \in \{1, \dots, n\}$ as

$$\tau_{r,j}^{(k)} := \phi(r|x_j, \Theta^{(k)}) = \frac{\alpha_r^{(k)} \mathcal{N}(x_j|\mu_r^{(k)}, \sigma_r^{2,(k)})}{\sum_{i=1}^2 \alpha_i^{(k)} \mathcal{N}(x_j|\mu_i^{(k)}, \sigma_i^{2,(k)})}$$

3. **Maximization step:** In step k , calculate the weight parameters $\alpha_r^{(k+1)}$ and $\alpha_2^{(k+1)}$, the means $\mu_1^{(k+1)}$ and $\mu_2^{(k+1)}$ and the variances $\sigma_1^{2,(k+1)}$ and $\sigma_2^{2,(k+1)}$ as follows:

$$\begin{aligned}\alpha_r^{(k+1)} &= \frac{1}{n} \sum_{j=1}^n \tau_{r,j}^{(k)} \\ \mu_r^{(k+1)} &= \frac{\sum_{j=1}^n x_j \tau_{r,j}^{(k)}}{\sum_{j=1}^n \tau_{r,j}^{(k)}} \\ \sigma_r^{2,(k+1)} &= \frac{\sum_{j=1}^n (x_j - \mu_r^{(k+1)})^2 \tau_{r,j}^{(k)}}{\sum_{j=1}^n \tau_{r,j}^{(k)}}\end{aligned}\tag{15}$$

Update the parameter set $\Theta^{(k+1)}$.

4. **Abort criteria:** Calculate the difference of the log-likelihood given by

$$\ln \phi(\mathcal{X}|\Theta) = \sum_{j=1}^n \ln \left[\sum_{i=1}^2 \alpha_i \mathcal{N}(x_j | \mu_i, \sigma_i^2) \right]$$

between $\Theta^{(k+1)}$ and $\Theta^{(k)}$. In case the difference is sufficient small the iteration can be aborted.

Threshold determination

The parameter set Θ contains the estimated mean μ_1 of the scores of the innocent-fingerprints, now denoted as μ_{inn} , and the estimated mean μ_2 of the scores of colluder-fingerprints, now denoted as μ_C . Hence the accordingly estimated parameters σ_1^2 and σ_2^2 are matched to the appropriate parameters for the variance of the innocent-fingerprints' scores $\sigma_1^2 = \sigma_{\text{inn}}^2$ and of the variance of colluder-fingerprints' scores $\sigma_2^2 = \sigma_C^2$. Note that, if applied in practice, these parameters come from the actual case of collusion attack. This is contrary to a priori computed expectation values, what is common sense in literature where the focus is on asymptotic (very large collusions) behavior of the code, see our explanations in chapter 7.2.

The crucial task is to suffice the selected bounds on false positive and false negative error, ε_1 and ε_2 . In literature, ε_2 is frequently set to 0.5, for reason of comparison, so it is here. Hence, the primary idea is to establish the estimated mean $\mu_2 = \mu_C$ of the second component of the mixture distribution as upper bound for the new dynamic threshold. Note that this choice of a threshold is motivated by the choice of Škorić *et al.* [121], where the probabilistic mean of the estimated colluder-fingerprints' scores was taken and this decision was followed ever after. On the other side, the threshold ought to be larger than the highest score among the innocent-fingerprints. Assuming a standard normal distribution an error of 1 is achieved at $\text{Erf}^{\text{inv}}(1 - \varepsilon_1)$. Mapped to the fingerprint construction analysis according to [121] this gives a lower bound for the threshold at $\Psi := \text{Erf}^{\text{inv}}(1 - \varepsilon_1) \cdot \sqrt{2} \cdot \sigma_{\text{inn}} + \mu_{\text{inn}}$. The corresponding proofs can be seen in [121], where this has been widely studied. This means, if holds $\Psi < \mu_C$, the threshold satisfying ε_1 and ε_2 lies in the corresponding interval $[\Psi, \mu_C]$. Results with the limits of this interval, i.e. for $Z = \mu_C$ and $Z = \Psi$ are presented in section 8.3.2. To suffice ε_1 and ε_2 any choice within this interval is feasible. In practice, the distributor or copyright holder has the choice, for instance, to select a 'conservative' setting of the threshold (i.e. $Z = \mu_C$) in order to be sure not to suspect

Table 15: Results for false positive rate (FP) and average number of colluders caught ($|C|$) with different threshold calculations Z (Z^* , GMM^+ , GMM^-), four different score functions and varying collusion sizes c while $c_0 = 15, 21$ and 31 after an interleaving attack.

c_0	c	Z	'Furon' [75]		'Škorić' [121]		'Oosterwijk' [86]		'Laarhoven' [62]	
			FP	$ C $	FP	$ C $	FP	$ C $	FP	$ C $
$c_0 = 15$	$c = 13$	Z^*	0	1	0.21	12.82	0.05	9.74	0.00	9.83
		GMM^-	0.32	13.00	0.22	12.82	2.17	13.00	0.34	13.00
		GMM^+	0	6.16	0.00	6.38	0.05	7.51	0.00	6.47
	$c = 15$	Z^*	0	1	0.22	13.89	0.05	7.07	0.00	6.04
		GMM^-	0.24	15.00	0.23	13.88	2.20	14.98	0.34	14.99
		GMM^+	0	7.44	0.00	7.08	0.13	8.87	0.00	7.47
	$c = 18$	Z^*	0	1	0.21	13.21	0.05	3.66	0.00	1.83
		GMM^-	0.28	17.76	0.21	13.11	2.29	17.66	0.36	17.73
		GMM^+	0	9.08	0.02	7.80	0.37	11.18	0.00	8.95
$c_0 = 21$	$c = 18$	Z^*	0	1	0.24	17.70	0.01	14.15	0.00	14.37
		GMM^-	0.36	18.00	0.24	17.70	1.78	18.00	0.33	18.00
		GMM^+	0	8.88	0.00	8.86	0.00	9.55	0.00	8.94
	$c = 21$	Z^*	0	1	0.23	19.06	0.01	10.11	0.00	8.99
		GMM^-	0.24	20.96	0.23	19.04	1.80	20.97	0.33	20.98
		GMM^+	0	10.32	0.00	9.96	0.03	11.46	0.00	10.47
	$c = 24$	Z^*	0	1	0.23	18.43	0.01	6.22	0.00	4.09
		GMM^+	0.52	23.68	0.23	18.32	1.84	23.72	0.33	23.79
		GMM^-	0	11.96	0.01	10.88	0.10	13.60	0.00	11.92
$c_0 = 31$	$c = 24$	Z^*	0	1	0.24	23.88	0.00	22.48	0.00	22.81
		GMM^-	0.40	24.00	0.24	23.88	1.22	24.00	0.32	24.00
		GMM^+	0	12.20	0.00	11.92	0.00	11.69	0.00	11.93
	$c = 31$	Z^*	0	1	0.24	27.52	0.00	15.32	0.00	14.01
		GMM^-	0.20	31.00	0.24	27.46	1.23	30.95	0.31	30.97
		GMM^+	0	15.28	0.00	14.85	0.00	15.72	0.00	15.43
	$c = 38$	Z^*	0	1	0.23	24.21	0.00	6.69	0.00	3.92
		GMM^-	0.28	37.16	0.22	23.79	1.29	36.90	0.32	37.03
		GMM^+	0	19.60	0.05	16.93	0.03	20.25	0.00	18.82

any innocents. Or he could select an 'offensive' setting (i.e. $Z = \Psi$), in order to catch as many colluders as possible.

8.3.2 Evaluation

Setting the threshold applying mixture models solely requires the actual scores of the current collusion attack. Its application does not depend on the digit model or on the alphabet size.

We tested our threshold calculation with the symmetric score function of Škorić *et al.* [121] and the interleaving score function of Oosterwijk *et al.* [86] we know from the afore chapters. In addition, to prove the adaptivity we tested it against the score functions of Meerwald and Furon

in [75] and *Laarhoven* in [62]⁵. We reason this choice by the eminent performance in practical settings of the iterative single accusation score function proposed in [75], and by the asymptotic optimality of the score function in [62]. Remember, the interleaving score function of [86] also proves asymptotic optimality.

The results are depicted in table 15. The code length was set to $m = 12,248, 22,912$ and $47,556$ for $c_0 = 15, 21$ and 31 respectively. The number of fingerprints was set to $n = 1,000$. For the error bounds we decided for $\varepsilon_1 = 1/n$ and $\varepsilon_2 = 1/2$. We ran 10,000 attacks for the symmetric score function [121] and for the interleaving score function [86] as well as for the score function proposed in [62] and per selection of c , each with Interleaving attack, minority vote and majority vote attack. Because they are very similar, only the results of the Interleaving attack are depicted here. Results for the minority vote attack and majority vote attack are provided in the appendix J. For several reasons – see below – the score function according to [75] was tested only against a series of 25 attacks for each setting.

The notation is as follows: FP stands for the average number of occurrences of the event of a false positive, i.e. an innocent that got accused. $|C|$ denotes the average number of colluders accused in every attempt. Hence, the error bounds are satisfied if holds $FP \leq n\varepsilon_1$ and $|C| \geq c\varepsilon_2$. Z^* refers to whether a threshold calculation according to [121] for the corresponding symmetric score function in [121], or to a threshold calculation according to [55] for the interleaving score function proposed in [86] and its counterpart recently published in [62]. Note that the computational complexity is comparably large for the single iterative score function of [75] due to the integrated estimation of the collusion attack. For a fair comparison to the other tested score functions regarding computational complexity, we stopped after the first iteration and accused the fingerprint with the highest score. That is why the false positive for Z^* is always zero and the number of colluders caught is always one. However, due to the discriminative nature of this score function, application of the GMMs already proved well. We selected Ψ and μ_C as lower and upper threshold for the GMM, i.e. $GMM^- = \Psi$ and $GMM^+ = \mu_C$. As an initial guess of the parameter set $\Theta^{(0)}$ we considered the first component to be standard normal distribution weighted with $\alpha_1 = (n - c_0)/n$. The second component is therefore weighted with $\alpha_2 = 1 - \alpha_1$ and initialized with the maximal score to be the mean and with a variance of two. This choice is reasoned by the large number of innocent-fingerprints' scores represented by the first component compared to the small number of colluder-fingerprints' scores of the second component. Thereby the initial guess is already aligned in the right direction.

The results achieved with mixture models show competitive ability compared to the thresholds that are tailored to its scoring function. In some cases, the results for the interleaving score function [86] fall behind, as we cannot satisfy the requested bound on ε_1 . Observations show that this is reasoned – for instance – by a skewness of the distribution of the scores. In cases for which the distribution is no longer Gaussian, i.e. for which the higher moments deviate from zero, the required error bounds might be hurt. A reasonable approach to manage this problem would be to use mixture models of skew-normal distributions.

The false negative error rate (FN) is not listed in table 15 because the required upper bound ($\varepsilon_2 = 0.5$) was satisfied in all attacks and configurations for all tested decoders ($\varepsilon_2 = 0.5$). However, with the ordinary threshold calculation Z^* the FN rate was up to 0.14 (for $c > c_0$) for the score function proposed in [62] and stayed below 10^{-3} for the interleaving score function [86], whereas the FN rate was zero in all parameter settings with GMM^+ and GMM^- . Hence,

⁵ This tests were conducted by means of the Matlab toolbox for Tardos codes provided by *Teddy Furon* <http://people.rennes.inria.fr/Teddy.Furon/website/software.html>

solely considering the false negative error rate, the threshold calculation via mixture models provides an improvement, while the other properties are roughly the same.

In summary, this approach allows application on any scoring function to separate innocents from colluders. The approach is independent of the conducted attack and of the fingerprint generation process. Moreover, the method does not need knowledge of the collusion strategy or its size, making it a universal threshold calculation. The results in table 15 show, that compared to ordinary threshold calculations especially tailored to the selected scoring functions, this method shows slightly improved robustness against outliers. Especially in cases in which the actual collusion size c differs from the maximum collusion size to be resistant against c_0 that was selected for the fingerprint generation, the threshold calculated via mixture models meets the selected error bounds longer than with the other thresholds.

8.4 Summary

We presented three approaches that optimize the error rates for the basis symmetric Tardos scheme ([114] and [121]). As not all parameters can be optimized at the same time, the different approaches all have their advantages and also their downsides.

The approach presented in section 8.1 employs a joint tracing algorithm following the idea of [74]. For evaluation, our approach utilized the fingerprint generation according to [11] described in chapter 7.2 and combines a single tracing step according to [121] and subsequently an enhancement of the joint tracing algorithm by [74]. The result was published in [10].

The approach from section 8.2, presents an enhancement of our idea of a correlative fingerprint generation from [11]. It introduces rank correlation coefficients into the fingerprint matrix that results in a better separation of innocent-fingerprints and colluder-fingerprints. The approach is based on our results in [71].

Our approach presented in section 8.3 proposes a universal dynamic threshold calculation applying mixture models. For instance, we are able to use the interleaving score function of [86] that is announced as asymptotical (very large collusions) optimal without the need of having knowledge about the attack strategy. The approach was published in [96].

False positive error bound: All approaches aim at reducing the false positive error rate, hence all results show significant improvements compared to their predecessors in literature.

False negative error bound: The joint tracing approach, section 8.1, provides false negative error rates that are far below the chosen upper bound ε_2 .

The correlative fingerprint generation, section 8.2 keeps equal or slightly improved error rates compared to [121] at significantly improved false positive rates.

The dynamic threshold using mixture models, section 8.3 shows error rates that are often-times (though only for smaller values of c_0) higher than for the fixed threshold according to the symmetric *Tardos* scheme [121]. The false negative rate is always close to the chosen upper bound ε_2 which, with respect to the other (improved) parameters, mirrors the high accuracy of the approach.

code length: The joint tracing algorithm is able to reduce the code length substantial, whether compared to the joint tracing algorithm of [74] for smaller value of c_0 or compared to [121] for larger values.

The other two approaches presented in this chapter do not affect the code lengths per se, they utilize the code length formula of the symmetric *Tardos* scheme by [121] or the optimal parameter choice according to [64], which is described in the appendix A.

number of fingerprints: The number of fingerprints can be chosen almost arbitrary, whether $n = 100$ or $n \geq 100.000$, the affect on the approaches stays negligible.

complexity: Compared to the single tracing of the symmetric *Tardos* scheme by [121], the joint tracing approach proposed in section 8.1 provides an increased complexity. Compared to other joint decoding approaches alike [74], the complexity is significantly decreased. Hence, our approach allows joint decoding tracing also for larger values of c_0 that other joint tracing algorithms cannot handle.

The other approaches proposed in this chapter also provide an increased complexity, because they also add mechanisms to the basis scheme of [121]. The amount of complexity added to these approaches remains negligible.

maximum chosen collusion size: We mentioned that our joint decoding approach is able to handle collusion sizes up to $c_0 = 30$, where sizes of $c_0 = 10 - 12$ is the practical maximum for other joint tracing approaches, e.g. [74].

The enhanced correlative matrix generation approach handles arbitrary values for c_0 , best results are attained for values larger than $c_0 \geq 20$.

The mixture model approach only works accurately for larger values of $c_0 > 15$. This is due to the statistical requirements on mixture models, demanding a certain number of samples for a reliable fitting of the components against the samples, i.e. the accusation scores. If the number for the second component ($=c_0$) is too small it is not possible to fit it against the curve of the colluder-fingerprints.

This chapter completes our elaborations on probabilistic fingerprinting codes. The major achievements are presented in this chapter. The joint tracing algorithm is a general enhancement of single tracing algorithms in case the complexity is not important. In practical settings where single tracing algorithms do not suffice, the joint tracing algorithm adopts the colluder tracing. The dynamic universal threshold calculation is not only independent of the score function but also independent of the attack strategy. This allows utilizing its improved threshold calculation for any probabilistic fingerprinting scheme.

9 Fingerprinting codes for database tracing

In this chapter we describe our approach to apply collusion secure fingerprinting schemes for hash databases. The approach was published in [14]. To the best of our knowledge, our concept is the first to combine fingerprinting and hash tables in order to trace unauthorized re-distribution of databases. Hash table databases are inter alia used for blacklisting and whitelisting, as discussed in chapter 3.7. Because of its potential for misuse of highly sensitive data, e.g. hashes associated to child-pornographic material, distribution of hash tables is very rare. The main administration is restricted to the different police departments (regional and nationwide). A reliable method to discourage misuse and re-distribution of hash tables is essential if you want to hand out these lists also to other parties, e.g. Internet providers or firewalls. These could benefit from the lists and detect illegal material and report its transmission to the corresponding police entity.

Approaches to watermark databases were already published by [3], [104] and [68] focusing on cryptographic hashes that do not allow any modification of the data. Other approaches that individualize hash tables by adding a salt were published by [98] and [17].

Our approach utilizes dummy hashes as fingerprint symbols. To individualize a hash table, the appropriate dummy hashes are blended into it. The blending process is the equivalent to the embedding process for media watermarking scenarios.

9.1 Construction of a collusion secure hash table

Individualizing hash tables using collusion secure fingerprinting schemes promises a new method satisfying the corresponding requirements regarding security, robustness and performance. To do so, the following (fingerprinting) parameters have to be chosen.

n : The number of fingerprints, the number of individual hash tables respectively.

m : The code length, which represents the number of dummy hashes embedded into each hash table.

c_0 : The maximum number of attackers\colluders the scheme promises resistance against.

ε_1 : The upper bound on the false positive error rate, bounding the probability of accusing innocents.

Fingerprint generation: Thereby the $m \times n$ fingerprint matrix \mathbf{X} is generated. For our approach we decided applying the fingerprint generation process according to [11]. The reason for that the good compromise between code length and error probabilities. As it was already presented in chapter 7.2, we will not recap it here. Note that all other fingerprint generation processes would serve as well, as long as they suffice the chosen fingerprinting parameters listed above.

Dummy hash creation: Next, before the embedding process starts, $2 \times m$ dummy hashes are generated once. These hashes can be built arbitrarily, the only requirement is an appropriate fitting within the entries of the original hash table. This means, without the knowledge of these dummies it must be impossible for attackers to detect them.

Attach fingerprint positions to dummy hashes: Each dummy hash $d_{i,\alpha}$ represents a (fingerprint) position i , $i \in \{1, \dots, m\}$ as well as its corresponding symbol $\alpha \in \{0, 1\}$. This means, for every column $i \in \{1, \dots, m\}$ of the fingerprint matrix \mathbf{X} there exists one

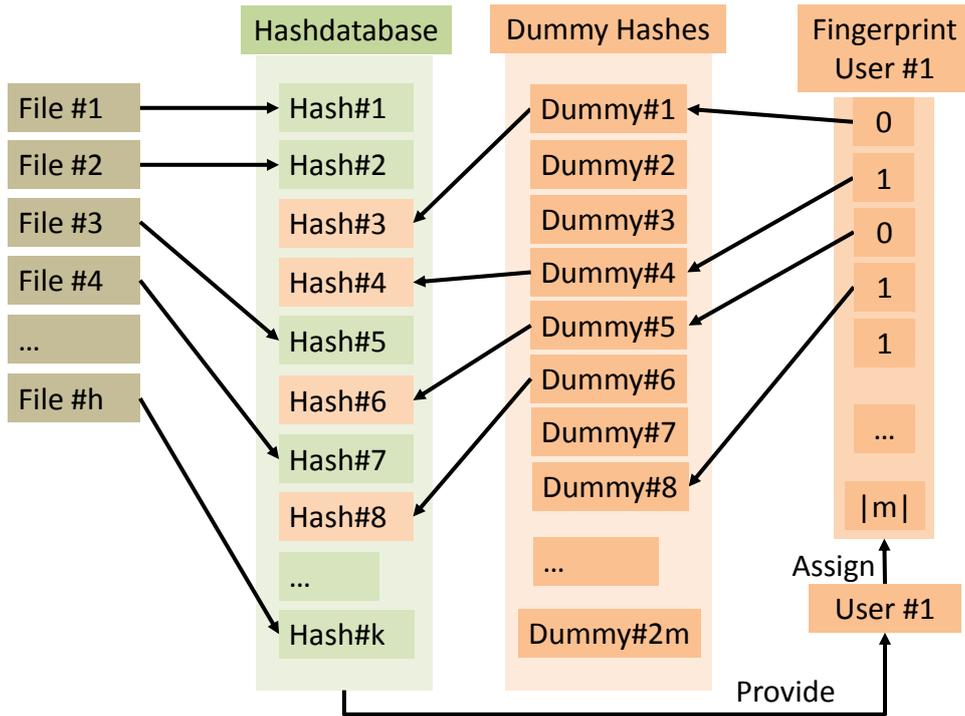


Figure 15: Hashtable Individualization Scheme

dummy hash representing the symbol '0', denoted by $d_{i,0}$, and one dummy hash representing the symbol '1', namely $d_{i,1}$. Hence, a fingerprint of dummy hashes is represented as $X_j = \{d_{1,\alpha}, d_{2,\alpha}, \dots, d_{m,\alpha}\}$.

Blending the dummy hashes: To individualize a hash database for user (or party) j , the dummy hashes that correspond to the j th row of the matrix \mathbf{X} , i.e. X_j , are blended into arbitrary positions in the hash database. The process is depicted in figure 15. The corresponding hash table database is individualized and from now on denoted as H_j .

This means, for example, for a fingerprint $X_j = (1, 0, 0, 1, 0, 1, \dots)$, the corresponding dummy hash set is $\{d_{1,1}, d_{2,0}, d_{3,0}, d_{4,1}, d_{5,0}, d_{6,1}, \dots\}$. The entries are blended into the hash table to make it the individualized hash table H_j . For security reasons, the dummy hashes should be blended uniformly distributed into the hash table and via random permutation. Otherwise a simple cropping attack, see section 9.3 could remove the dummy hashes.

9.2 Detection and tracing

The security of the scheme is based on the list of dummy hashes and their correct allocation. Only the holder of this list is able to reconstruct the correct message out of the hash table. Here, the list represents the function of the secret key in media watermarking scenarios.

To build the manipulated fingerprint, we filter all dummy hash entries of the unauthorizedly distributed hash table that can be found. To reconstruct the corresponding fingerprint, we order the dummy hash values in the correct way as prescribed by the list. The resulting binary code represents the fingerprint. In analogy to the other chapters it is also denoted as y .

With the reconstruction of y , the tracing operates in analogy to other tracing algorithms, for example those presented in this thesis, chapters 4.5, 7.2, 7.4, 8.1, 8.3. Each position of the ma-

nipulated fingerprint y is compared to each corresponding fingerprint position of all fingerprints in matrix \mathbf{X} in order to calculate the corresponding accusation scores. The only requirement on the selection of the tracing algorithm is to match the selection of the fingerprint generation process. For reasons of generality, we decided for the symmetric Tardos tracing algorithm according to chapter 4.5. Finally, all users whose fingerprint scores exceed the predefined threshold Z are considered to be part of the manipulation and output by the tracing algorithm.

9.3 Attack models

The attack models for the hash table fingerprinting scenario differ from the attacks presented in chapter 4.6.

In order to conduct a collusion attack, the attackers compare their databases and find some hash entries that do not appear in all colluders' databases. These hashes obviously belong to the number of dummy hashes utilized for the database individualization and therefore are called *detectable dummy hashes* (DD-hashes) in the following. However, a serious number of dummy hashes is not detected due to their appearance in each colluders' database. These are called *undetected dummy hashes* (UD-hashes). The union set of the original hashes and the UD-hashes is denoted as *intersection set*. The mean number of UD-hashes corresponding to an arbitrary set of c colluders is

$$|\text{UD-hashes}(c)| = \frac{2\Gamma[\frac{1}{2} + c]}{\sqrt{\pi}\Gamma[1 + c]} \cdot m,$$

where $\Gamma[z] := \int_0^{\infty} e^{-t} t^{z-1} dt$ represents the *Euler Gamma Function*, see [2]. The mean number for DD-hashes is given by

$$|\text{DD-hashes}(c)| = m - |\text{UD-hashes}(c)|.$$

For example, in case of $c_0 = c = 10$ colluders, 35% of the dummy hashes are undetectable. With these 35%, sufficient information is provided for the detector to allow successfully tracing back to the colluders.

All attack models are analyzed under the condition $c \leq c_0$ and based on the assumption that each attacker contributes approximately equal parts to the attacked database. For the case the attackers contribute unbalanced parts, the detector finds much easier at least one of the malicious users. In the following we list some apparent attacks.

Intersection attack: For an intersection attack the colluders discard all DD-hashes and take the intersection set of all their hashes for the attacked database. The security of the proposed scheme then only relies on the undetected dummy hashes.

Cropping attack: To avoid being traced back, instead of relaying the whole database, for this attack the colluders distribute only parts of it. In this work, the assumption is taken that cropping the database reduces the number of UD-hashes proportionally to the whole database. This means, in case the colluders select only 1/2 of the original database (whether of the intersection set or of the union set of all colluders' hashes) for distribution, the number of UD-hashes would be reduced by the factor 1/2 as well.

Minority adding: In order to set the tracers on the wrong track, the colluders may also add some of their DD-hashes to the distributed version.

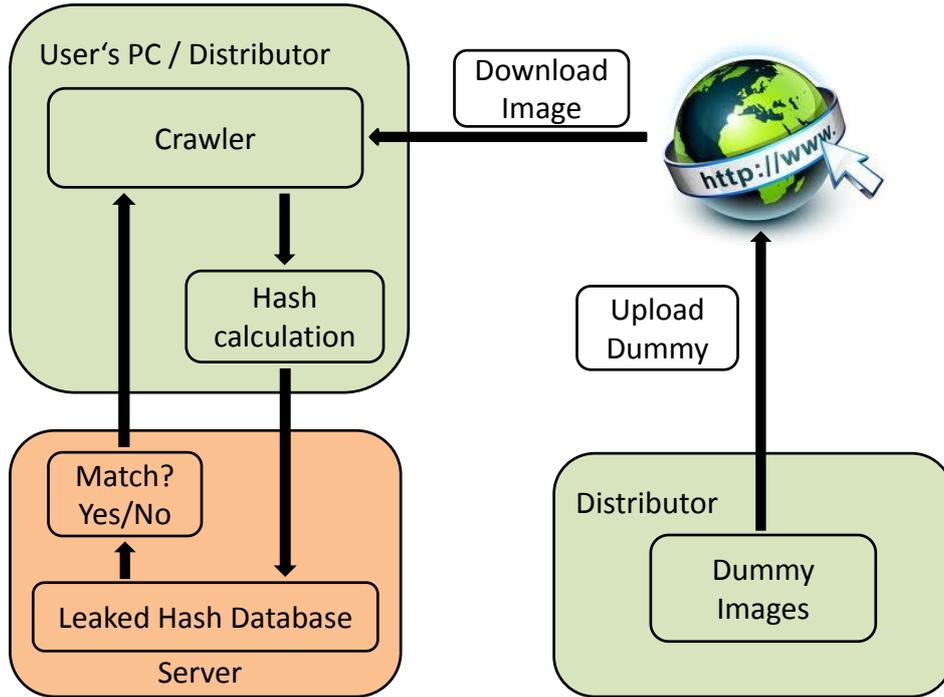


Figure 16: Leakage detection without access to the hash database

The case, where both, colluder a 's DD-hash d_a and colluder b 's DD-hash d_b are added to the list to be distributed and in addition, d_a represents a '0' and d_b represents a '1' of the same fingerprint position, this position is denoted as *double position*.

To avoid the possibility of double positions within the integrated fingerprint of the attacked database, only (some of) the DD-hashes are added which appear in less than $\lfloor c/2 \rfloor$ times in the colluder's databases.

Majority adding: Refers to the attack that arbitrarily adds some of the DD-hashes which appear in more than $\lfloor c/2 \rfloor$ times in the colluders' databases. In this way the possibility of a double position is avoided as well.

Full minority adding: In the *full minority adding* attack, the colluders add all DD-hashes to the list to be distributed which appear less than $\lfloor c/2 \rfloor$ times in the colluders' databases.

Full majority adding: For the *full majority adding* attack, the colluders simply add all DD-hashes which occur more than $\lfloor c/2 \rfloor$ times.

9.4 Evaluation and discussion

To evaluate our concept we tested with different numbers of fingerprints n , different numbers of colluders $c_0 = c$ and different number of dummy hashes m per individualized hashtable. Table 27 presents the resulting false positive errors FP , and the corresponding false negative errors FN . Note that the scheme is independent of the number of hashes in the database.

Note that in a reduced hash table, for example, if only 1/2 of all entries are distributed, the number of dummy hashes to be found, the fingerprint positions respectively, will be reduced accordingly. The result is a discontinuous fingerprint for which the tracing algorithm needs to be adjusted.

Table 16: Comparison of Code length, false positive error (FP) and false negative error (FN) at small c_0

attack strategy	number of hashtables	c_0	dummy hashes	FP	FN
intersection	100	5	760	0	0
	5000	5	854	0	0
	1000	10	2672	0	0
	5000	15	7679	0	0
cropping 1/2	100	5	568	0	0
cropping 1/4	100	5	760	$35 \cdot 10^{-6}$	0
cropping 1/2	5000	5	954	10^{-6}	18.8%
cropping 1/4	5000	5	1908	0	8.6%
cropping 1/2	1000	10	2672	10^{-6}	0
cropping 1/4	1000	10	5344	0	3.6%
cropping 1/2	5000	15	7679	0	0
cropping 1/4	5000	15	15358	0	3.3%
minority adding	100	5	760	10^{-6}	0
	5000	5	854	0	0
	1000	10	2672	$32 \cdot 10^{-6}$	0
	5000	15	7679	0	0
majority adding	100	5	760	0	0
	5000	5	854	0	3.3%
	1000	10	2672	$38 \cdot 10^{-6}$	0
	5000	15	7679	0	0

A reduction of the error rates can be achieved easily by increasing the number of dummy hashes. The higher the number of dummy hashes is set, the lower will become the error rates. Here an appropriate trade-off between robustness against attacks and capability for adding hashes needs to be found.

Other attacks such as *mixture*, *mix and match*, *subset alteration* and *hash adding* are not specified in our evaluation, because the proposed code is resistant against these attacks by construction.

Following the example of blacklisting, the list consists of hashes associated to child-pornographic content. It is leaked for crawler optimization purposes. The perpetrators do not emit the hash database, but the hash comparison must be rendered on a server hosted from the perpetrators. To trace back whose blacklist it is, the distributor uploads to the Internet those dummy images that worked as templates to generate the dummy hashes. Thereby the crawler searches for the uploaded dummy images starting from the uploading page. A report from the crawler shows the set of those dummy images which are blended in the leaked database and out of these the distributor is able to identify the perpetrator(s). Figure 16 depicts this scenario.

9.5 Summary

This chapter presents a new method to apply fingerprinting codes in order to trace back leakage of hash table databases. To the best of our knowledge this is the first of its kind. Individualization of a hash table for different users is done on the fly by adding appropriate dummy hashes to arbitrary positions of the existing hash database. Each of the dummy hashes represents one bit-symbol of the fingerprint and is calculated only once in advance. The proposed code is independent of the size of the hash database.

The concept suffices the following properties.

false positive error bound: The false positive error rate remains far below the chosen rate ϵ_1 , even for a reduced hash table of only 1/4th of the original.

false negative error bound: Similar to the false positive error rate, the false negative error rate remains very low. The dependency to the reduction of the hash table is yet visible (see table 16).

code length: The number of dummy hashes is depending on the chosen fingerprinting scheme. In general, contrary to most watermarking scenarios, the available payload is not challenging in this scenario.

number of fingerprints: Alike most *Tardos Codes*, the number of fingerprints has only minor impact on the scheme.

complexity: Beside the typical efforts for the fingerprint generation process and the tracing algorithm, the additional effort remains linear and therefore is negligible.

maximum chosen collusion size: Whether the chosen bound on the maximum collusion size to be resistant against is small or comparably large, an appropriate fingerprinting scheme can be found. However, we do not expect collusions larger than $c_0 = 15$.

The proposed approach is a simple example to show the potential of fingerprinting codes for hash table databases.

Indeed, there is room for more sophisticated approaches, adjusted to this application of databases. For example, since the dummy hashes are not somehow connected to the symbols of the fingerprints, the use of higher alphabets is capable to embed more information for the same number of blended dummy hashes. However, this amount of more information also induces yet a higher variety of different attack models.

The remaining challenge is the ability to crop the hash table and re-distribute only fractals of it. In case cropping and re-distributing only a fractal is a realistic option for attackers, the selected fingerprinting scheme and its parameter selection ought to be adjusted accordingly.

10 Application scenarios supporting fingerprinting codes

In this chapter we describe the concepts enabled by collusion secure fingerprinting codes for the scenarios described in chapter 3. Therefore we will rely on the different properties and characteristics analyzed in chapter 4 and consider the requirements of the corresponding scenario as described in chapter 5, in order to attach the fingerprinting scheme to the appropriate application scenario.

This way we revert to the initial scenarios and attach fingerprinting codes to receive enhanced scenarios. The enhancement is mainly restricted to the requirements regarding security that we identified throughout this thesis.

To evaluate the results we take on the table(s) given in chapter 5. Based on the identified requirements we show which of the fingerprinting schemes explained in chapters 6 through 9 suffice which requirements and thereby which application scenario. The tables in the following sections specify if the corresponding requirement is satisfied with a '✓'. A '—' denotes the case in which a requirement remains unrealized.

10.1 Fingerprinting codes for promotional application scenarios

The promotional application scenario presents a classical scenario for the application of watermarks and fingerprinting schemes. To efficiently counter collusion security, the application of fingerprinting schemes is essential. As summarized in table 4 of chapter 5 the required maximum expected collusion size c_0 is comparably small. A collusion of more than two or three participators is rather improbable.

The 2-secure fingerprinting scheme as well as the 3-secure fingerprinting scheme both introduced in chapter 6 are optimized regarding error probabilities and complexity, at the same time provide shortest code lengths. They suffice the requirements for the promotional application scenario and are capable to build the bridge between theory of optimal codes and their application in practice.

In addition, the ranking search approach presented in chapter 7.1 and further the combination of the discrete fingerprint generation process, chapter 7.3 and the interleaving score function tracing scheme with dynamic threshold presented in chapter 7.4 may assist in case the expected number of colluders exceeds $c_0 > 3$. The combination is easily possible as the one approach represents a fingerprint generation process while the other represents a tracing algorithm and they are developed in a way that they can be attuned to each other.

The following requirements are satisfied:

maximum expected collusion size c_0 : Our zero false positive 2-secure fingerprinting scheme (chapter 6.1) is solely resistant against two colluders, analogously the 3-secure fingerprinting scheme (chapter 6.2) is resistant against three colluders. This suffices the promotional application scenario as described in chapter 5.1. Otherwise the other two approaches listed in table 17 (chapter 7.1 and the combination of 7.3 and 7.4) stand in yet with limited possibilities.

number of fingerprints n : The 2-secure approach is limited in its numbers of fingerprints. It is optimized for $n \leq 1.000$. For higher values of n , regarding the other parameters the other approaches may serve better. The ranking search approach, if used as a proper tracing algorithm, also shows its best results if n is likewise small. The effect for the other two approaches listed in table 17 is negligible.

Table 17: Fingerprinting solutions for the promotional application scenario

Promotional application scenario	Fingerprinting scheme solutions			
Requirements on:	<i>2-secure</i>	<i>3-secure</i>	<i>ranking search</i>	<i>discrete distribution + dynamic threshold</i>
Max. expected collusion size c_0	≤ 2	≤ 3	2-6	6-15
Number of fingerprints n	(\checkmark)	\checkmark	(\checkmark)	\checkmark
Code length m (M)	\checkmark	\checkmark	(*)	(*)
False positive error rate FP	\checkmark	\checkmark	\checkmark	\checkmark
False negative error rate FN	\checkmark	\checkmark	\checkmark	\checkmark

code length m : The selected code length depends on the available payload M provided by the actual media cover. If m exceeds M , the fingerprinting scheme cannot be applied. Also, in media watermarking scenarios it is recommended to embed the fingerprint several times to increase robustness and security. Thus, the approach providing the shortest code lengths is beneficial. The 2-secure approach provides the shortest code length of (depending on the other parameters) less than $m \leq 100$ bits. The 3-secure approach and the ranking search approach provide similar short code lengths for $c_0 = 3$ that are around $m \approx 100$ (also depending on the other parameters). The typical media covers in the promotional application scenario are capable to embed fingerprints of these lengths. However, in case of a choice for c_0 larger than 3, the corresponding code lengths for the ranking search approach and the discrete generation combined with the dynamic threshold approach may exceed the available bounds for media cover with limited payload, for example short audio tracks.

False positive error rate FP : The 2-secure approach provides a provable zero false positive rate. Therefore its output is hardly contradictable. The 3-secure approach provided false positive rates far below the required lower bound ε_1 . The other two approaches also provide error rates that stay below the lower bound ε_1 , although, for some parameter choices, they come very close to it.

False negative error rate FN : For this scenario we required at least a false negative error rate of 0.5. All approaches are capable to suffice this requirement, in most cases they stay far below this bound.

The ability of the fingerprinting schemes developed in this work to satisfy the identified requirements for this scenario were documented in the corresponding evaluation sections. More information regarding the evaluation is provided in the appendix, C, D, F and G.

10.2 Fingerprinting codes for online shops

The online shop scenario requires different adjusted fingerprinting schemes for the different media covers and different requirements on the fingerprinting parameters. Table 4 in chapter 5.6 tells us that the maximum expected collusion size as well as the available payload is of medium size whereas the number of fingerprints oftentimes is comparatively large.

We also found exceptions. We identified a small value for c_0 only for Ebooks. Therefore, in case the other parameters fit as well, the fingerprinting schemes suitable for the promotional

Table 18: Fingerprinting codes for the online shop application scenario

Online shop scenario	Fingerprinting scheme solutions				
Requirements on:	<i>ranking search</i>	<i>correlative matrix</i>	<i>discrete generation</i>	<i>dynamic threshold</i>	<i>joint tracing</i>
Max. expected collusion size c_0	2-6	arbitrary	6-15	6-15	arbitrary
Number of fingerprints n	-	✓	✓	✓	✓
Code length m (M)	✓	-	(✓)	(✓)	✓
False positive error rate FP	✓	✓	✓	✓	✓
False negative error rate FN	(✓)	(✓)	✓	✓	✓

application scenario may also satisfy the requirements for Ebooks and can be applied without compunction. However, the large number of fingerprints probably prevents from applying the 2-secure fingerprinting approach.

As already mentioned in chapter 5.6, we excluded the online shop distribution formats video games and streaming media and sorted them into the general video game application scenario as it better fits their requirements.

The other media types listed in the online shop scenario can be protected applying the following approaches introduced in chapters 7 and 8.

- **Ranking search approach**, chapter 7.1
- **Correlated fingerprint matrix generation for the continuous distribution**, chapter 7.2
- **Correlated fingerprint matrix generation for the discrete distribution**, chapter 7.3
- **Dynamic threshold approach**, chapter 7.4
- **Joint tracing approach**, chapter 8.1

Table 18 depicts in how far the requirements for the online shop are satisfied by the approaches listed above.

maximum expected collusion size c_0 : As the online shop requires collusion security against collusions of medium sizes, the ranking search approach (applied as a sovereign tracing algorithm) may suffer some challenges in case c_0 exceeds 6. The approach of correlative fingerprint matrix generation for continuous distributions is not affected by the choice of c_0 . Its discrete version as well as the dynamic threshold approach provide suitable results in a range of $c_0 \in \{6, \dots, 15\}$. The joint tracing algorithm may assist in case larger values for c_0 are demanded, but also for smaller values in case complexity is not an issue, the joint tracing algorithm performs well.

number of fingerprints n : Because the required number of fingerprints was identified as large, if the other parameters are selected appropriate for this scenario, the ranking search approach probably fails to act. Hence this approach is reduced to its function as a filter reducing the complexity before subsequent tracing algorithms start. The other approaches in table 18 are not noteworthy affected by the choice of n .

code length m : In case it suffices the choice of n , the ranking search approach provides comparably short code lengths, as already mentioned in section 10.1. The approach providing a

correlative generation for continuous distributions though significantly reducing the code length compared to its predecessors (see chapter 7.2), it cannot keep up with the other approaches of chapters 7 and 8.1. For this reason table 18 reflects a '–'. The other three approaches provide significantly reduced code lengths while fixing the other parameters. As mentioned in the last section, the discrete generation approach can be combined with e.g. one of the two tracing algorithms, dynamic threshold or joint tracing, to further reduce m . However, for the media covers that provide only small amount of payload, i.e. music tracks and images, the corresponding fingerprints possibly remain too long for successful embedding. For those media types the only chance of collusion security at all is to revert to appropriate fingerprinting schemes from the promotional application scenario.

False positive error rate FP : All approaches are designed in order to reduce the false positive error rate while the other parameters are at least fixed. All approaches stay below the required lower bound ε_1 given the other parameters as required.

False negative error rate FN : For some parameters the false negative error rate in the ranking search approach scratches along the lower bound ε_2 . As we find the false negative error not as important as the false positive error, table 18 depicts a '✓' in brackets. The continuous generation approach focuses on reducing the code length and the false positive error rate by inter alia calculating an adjusted threshold for isolating colluder-fingerprints from innocent-fingerprints. This involves a slight increment of the false negative error rate, sometimes exceeding ε_2 (in case of very specialized attacks as the mutual information attack or the greedy attack both described in chapter 4.6). The same holds for the discrete generation approach and the dynamic threshold for the same reason. However, they are affected significantly less, so that the requirement on the false negative error rate remains clearly fulfilled. The joint decoding approach provides a false negative error rate that stays considerably below the demanded upper bound ε_2 .

All in all we recommend the ranking search approach in case relatively small values of c_0 (≤ 6) are accepted or if the available payload prescribes a corresponding small value for c_0 . For values of c_0 between 6 and 15 the dynamic threshold approach or rather the combination of the discrete distribution and the dynamic threshold approach are the fingerprinting schemes to be preferred for the online shop scenario. For the case that the desired value for c_0 is larger, the joint tracing approach (e.g. combined with correlative fingerprint generation) can be selected instead. Also for smaller choices of c_0 , the results can compete with the other approaches. Its only downside is the increased complexity. As already mentioned in chapter 8.1 we further improved the results by combining the joint tracing with the correlative fingerprint generation. However, if the prescribed payload provided by the underlying watermarking basis prevents from applying the just discussed approaches, the only chance to provide collusion security might be the 2-secure approach or the 3-secure approach we suggest for the promotional application scenario.

10.3 Fingerprinting solution for digital cinema

In chapter 5.3 we identified the requirements for the digital cinema application scenario. The scenario strongly depends on the media cover and on the content, i.e. is the movie a large blockbuster production for worldwide screening or is it a local independent movie production. We committed ourselves to *medium* sized maximum expected collusion size c_0 and available payload M , which prescribes the code length m . The number of cinemas screening the movie, i.e. the number of fingerprints, is assumed as *large* and the error rates are kept as in the other

Table 19: Fingerprinting codes for the digital cinema application scenario

Digital cinema	Fingerprinting scheme solutions			
Requirements on:	<i>ranking search</i>	<i>discrete generation</i>	<i>dynamic threshold</i>	<i>joint tracing</i>
Max. collusion c_0	2-6	6-15	6-15	arbitrary
No. of fingerprints n	-	✓	✓	✓
Code length m (M)	✓	✓	✓	✓
False pos. error FP	✓	✓	(✓)	✓
False neg. error FN	(✓)	✓	✓	✓

scenarios for comparison reasons. This prescription regarding the fingerprinting parameters leads to the following fingerprinting schemes.

- **Ranking search approach**, chapter 7.1
- **Correlated fingerprint matrix generation for the discrete distribution**, chapter 7.3
- **Dynamic threshold approach**, chapter 7.4
- **Joint tracing approach**, chapter 8.1

Only in case the number of screening cinemas, that is the number of fingerprints, is comparably low, the *Ranking search approach* proposed in chapter 7.1 might be an option as well.

In table 19 we depict the validity for the digital cinema application scenario for the approaches mentioned above. Due to the comparatively large payload and because we expect only medium sized collusions, the requirements are clearly satisfied.

maximum expected collusion size c_0 : As unauthorized re-recording in cinemas is aroused by (semi-) professional groups, acting in several places a movie is screened, a medium sized value for c_0 is recommended. Hence, the ranking search approach (applied as a sovereign tracing algorithm), which can only be applied if the choice for c_0 is relatively small, is only fairly suitable for this scenario.

The correlated fingerprint matrix generation for the discrete distribution performs best against medium sized collusions. This also holds for the approach providing the dynamic threshold calculation. Both approaches are well suited to be combined, inducing a further improvement of the scheme.

In case of a *medium* sized value for c_0 , the joint decoding approach is well suited.

number of fingerprints n : Similar to section 10.2, application of the ranking search approach solely is reasonable in scenarios for which n is comparatively small.

The other approaches in table 19 are not (significantly) affected by the number of fingerprints.

code length m : If possible to apply the ranking search approach, it provides best results regarding the code length.

The approach providing a correlative fingerprint generation using a discrete distribution reduces the code lengths compared to its originating scheme. But the impact is relatively

small. If combined with the dynamic threshold calculation, however, the resulting scheme achieves significant reduction. That is because of the possibility to apply the interleaving score function.

The joint tracing approach provides best results regarding the code length.

False positive error rate FP : All approaches listed in table 19 clearly fulfill the false positive error rate. This is because we handled ε_1 as the crucial parameter to be satisfied. Also, compared to the other application scenarios, the false positive error rate may become loose for the digital application scenario.

False negative error rate FN : We already mentioned, that the ranking search approach (if applied as sovereign algorithm) for some parameter settings might exceed the bound on the false positive error rate. For this reason, the \checkmark in table 19 is set in brackets. However, in the digital cinema application scenario, most of these parameter settings do not occur.

As mentioned before, the approach providing a correlative fingerprint generation process according to a discrete distribution as well as the approach introducing a dynamic threshold for the tracing algorithm stay just below the required upper bound for ε_2 .

The joint tracing algorithm achieves significantly low false negative error rates.

We showed that the requirements for the digital cinema application scenario are clearly satisfied by the proposed fingerprinting schemes. Given a proper watermarking basis, the proposed schemes are capable to provide reasonable indications of the cinemas where unauthorized re-recording occurred.

10.4 Fingerprinting solution for boxed video games

Video games are a new scenario for the application of watermarking as we discussed in several publications during the last three years ([12], [118], [13], [70]). From its beginning, in both, our opinion but also in the opinion of all developers or publishers we talked to, collusion security is clearly required. In case a game should be protected by watermarking technology, the appropriate fingerprinting scheme must be employed.

In chapter 5.4 we identified the requirements on the fingerprinting parameters c_0 , n as large. The corresponding available payload M allows large code lengths m (table 4). This holds for large boxed video game productions, for smaller productions, i.e. in the independent sector, obviously smaller parameter values may suffice as well.

Our approaches proposed in chapter 8 suffice well these requirements, see table 20.

- **joint tracing algorithm**, chapter 8.1
- **enhanced correlative fingerprint generation**, chapter 8.2
- **universal threshold using mixture models**, chapter 8.3

They have in common, that they focus on larger expected collusion sizes and aim to reduce the false positive error rate compared to predecessors. The approach for the correlative fingerprint generation considers the generation of the fingerprints, while the other two approaches are pure tracing algorithms. The joint decoding tracing algorithm works with, as the name says, a joint accusation score function, i.e. uses tuples of fingerprints to calculate the scores, whereas the dynamic threshold approach considers an optimized and dynamic threshold calculation independent of the attack strategy and the selected score function.

Table 20: Fingerprinting codes for the boxed video game application scenario

Boxed video game scenario	Fingerprinting scheme solutions		
	<i>joint tracing</i>	<i>enhanced correlative fingerprint generation</i>	<i>universal threshold</i>
Max. expected collusion size c_0	arbitrary	>15	>15
Number of fingerprints n	✓	✓	✓
Code length m (M)	✓	(✓)	(✓)
False positive error rate FP	✓	✓	✓
False negative error rate FN	✓	(✓)	✓

Table 20 depicts the achieved results for the requirements of the boxed video game scenario. Because of the huge amount of payload that allows comparably long code lengths, also the other requirements can be satisfied by the proposed approaches. This is proven via huge testbeds already presented in chapter 8, or in the appendix H, I and J.

maximum expected collusion size c_0 : As the game industry faces professional release groups attacking their protection, there is high potential for large collusions. The joint tracing approach is capable to resist large collusions, but can be applied for smaller c_0 as well. The enhanced correlative fingerprint generation as well as the approach proposing the dynamic threshold calculation via mixture models may be applied for small values as well, but show their excellence for value larger than c_0 due to statistical properties.

number of fingerprints n : Large productions obviously target large customer bases, the corresponding number of fingerprints needed is large. All three approaches are capable to handle large values > 1.000.000.

code length m : The joint tracing approach provides significantly reduced code lengths, suitable also for large c_0 . The other two approaches have no direct influence on the code length, therefore the ✓ is put in brackets. As both approaches optimize (almost all of) the other parameters, a reduction in the code lengths is plausible for them as well.

False positive error rate FP : The video game community probably will not forgive an event in which an innocent user is accused. As all approaches show a strongly reduced false positive error rate, they suit this requirement as well.

False negative error rate FN : The false negative error rate for the joint tracing approach is significantly below the selected upper bound ε_2 . For the other approaches we decided to additionally sum up the average number of colluder-fingerprints output by the corresponding tracing algorithm. This is because the false negative error rate was met at all time. The received results lead to a false negative error rate smaller than with the approaches in literature we compared to.

The approaches of chapter 8.2 and 8.3, i.e. the correlative fingerprint generation and the tracing algorithm with dynamic threshold, have successfully been combined and tested. The results can be seen in the appendix J. The joint tracing approach was also tested in combination to our fingerprint generation approach presented in chapter 7.2, the results were already posted in 8.1 or in the appendix H. Its effect on the accusation scores turned out to be advantageous for the

joint tracing approach compared to the other approaches in literature, as can be seen in tables 12 and 27.

Given an appropriate watermarking basis, application of the approaches promise to provide the desired protection for the boxed video game application scenario.

10.5 Fingerprinting solution for hash table databases

Our approach for hash table fingerprinting presented in chapter 9 differs from the other approaches because no media watermarking basis is required to embed (and detect) the fingerprints. Moreover, this scenario does not employ a new fingerprinting scheme but exemplarily utilizes the scheme presented in chapter 7.2. Any other fingerprinting scheme can be applied instead as well. Following the course of this chapter, we list the approaches presented in this thesis that are appropriate to fit the requirements as identified in chapter 5.

- **Ranking search approach**, chapter 7.1
- **Correlated fingerprint matrix generation for the continuous distribution**, chapter 7.2
- **Correlated fingerprint matrix generation for the discrete distribution**, chapter 7.3
- **Dynamic threshold approach**, chapter 7.4
- **Joint tracing algorithm**, chapter 8.1
- **Enhanced correlative fingerprint generation**, chapter 8.2
- **Universal threshold using mixture models**, chapter 8.3

These are the approaches presented in chapter 7 and in chapter 8. The approaches in chapter 6 are not appropriate, because the identified value for c_0 was set as *medium*. All other approaches suit this requirement. The number of fingerprints remains comparably small, because the number of users or parties that are intended for an individualized hash table remains small. For this reason, all approaches listed may serve well. The crucial limitations for fingerprinting codes in general is the limited payload, i.e. the code length m . However, in this scenario the payload is comparably large, hence it is no challenge for the presented approaches. We abstain from posting a corresponding table, as all entries would be filled with a \checkmark anyway.

The hash table application scenario is an example for the employment of fingerprinting codes apart from the media watermarking scenario. The fingerprinting schemes presented in chapter 7 and in chapter 8 are capable to provide the desired protection as identified in chapter 3.7 and 5.

11 Open challenges and opportunities

In the course of our work on fingerprinting codes, we proposed solutions for several scenarios in the media watermarking sector. In our opinion, some solutions promised concepts or approaches worthwhile to proceed research. In this chapter we stress open challenges and formulate required future prospects regarding the research of fingerprinting codes.

11.1 Toolbox containing all recent and state of the art fingerprinting codes

There exists a magnitude of different approaches for collusion secure fingerprinting codes. As discussed in this thesis, many approaches that are optimal in theory lack of applicability in practice, e.g. [121], [86]. Though, we argue that some of these approaches require only small modifications or adjustment for a proper integration into practical applications. This holds true for the approach we presented in chapter 7.4. It was based on the theoretical optimal approach of [86]. Applying this approach requires knowledge of the – generally unknown – collusion strategy. To enable application in practice, we modified the approach and abolished the dependency of the collusion strategy.

There is a chance that for specific parameter settings (and) in some fringe scenarios, other yet unconsidered approaches outreach those approaches of the main focus (which have proven optimality from a general perspective). These unconsidered approaches have not found their way into practice.

This emphasizes the goal of the last chapters: There does not exist one single best fingerprinting code. Different scenarios enforce different fingerprinting codes, and the question to solve is, which fingerprinting code suits best the current specific application scenario. To solve this question, we suggest setting up a general junction library containing all fingerprinting codes available and employing the corresponding algorithms in a comprehensive toolbox. The toolbox ought to provide researchers and especially distributors of fingerprinting codes the possibility to choose a parameter setting as input (available payload, collusion size to be resistant against, error probabilities, etc.) and the output should be the corresponding optimal – in terms of best suited – fingerprinting code.

A first approach in this direction was initiated by *Teddy Furon* and his group at Inria, France. They provided a MATLAB-toolbox containing some of their fingerprinting codes as open source download ⁶. Expanding this toolbox with (all the) other fingerprinting codes that are publicly available appears as a reasonable starting point. On the downside is that, even though the toolbox is open source, MATLAB per se is not. It requires acquisition of at least a basic license. For the use of additional MATLAB-tools further licenses are required.

11.2 Higher alphabets for watermarking basis and fingerprinting codes

In order to reduce the code lengths of the fingerprints, many approaches utilize higher alphabets. This means, more symbols admissible to select for a fingerprint position allows to embed the same amount of information into fewer fingerprint positions, e.g. [121], [86], [8], [7].

In practice, fingerprinting codes using higher alphabets cannot evoke any advantage, because – to the best of our knowledge – the watermark embedding requires downscaling the fingerprint

⁶ <http://people.rennes.inria.fr/Teddy.Furon/website/software.html>

from higher alphabets to a binary again. This holds in particular for the watermarking algorithms developed and provided at Fraunhofer SIT: The watermarking embedding algorithm modifies the media cover to embed watermark information, see chapter 2.4. Modification is done by changing pixels of an image or video or altering audio samples, or modifying the respective frequency coefficients, or else. To embed a '1' or a '0' the relation of two groups of pixels, samples, frequency coefficients, etc., of equal weight is changed into the one direction or the other. In other words, the groups are modified in a way that the weight of the one group is heavier than the weight of the other group by a distinct amount. This way, there are only two states possible implying that the watermarking algorithm can only handle binary symbols. For that reason we concentrated on binary fingerprinting codes.

A watermarking algorithm that is able to efficiently embed higher alphabets would allow applying fingerprinting codes with higher alphabets. The outcome would be a significantly reduced fingerprint code length. However, remains in question, if the corresponding watermark message to be embedded actually requires fewer space in (less modification of) the cover medium, than what is required to embed the same amount of information with binary fingerprints and by binary watermarking applications.

In the procedure of this thesis we developed an idea to embed fingerprints with higher alphabets for audio files. The embedding algorithm used at Fraunhofer SIT operates on frequency coefficients that have been transformed from sample values per audio frame. In lieu of creating the two groups of a certain approved number of frequency coefficients, we could build, for instance, three groups of the same (number of) frequency coefficients. Three groups (instead of two groups) implies more states of relation of the groups. More states of relation implies more different symbols for one position, i.e. higher alphabets. The optimal way setting these groups in relation to each other is an open question. Also the optimal parameter setting for the embedding and detection is not found yet. We need to evaluate corresponding loss in robustness and transparency using three groups instead of two. Developing a concept and evaluating the benefit of higher alphabets for audio is an announced topic we plan to research in the near future.

Further research is needed in the following positions:

Watermark embedding of higher alphabets: Is it possible to develop/employ watermarking algorithms allowing the use of higher alphabets in an efficient way? To answer this question, the watermarking embedding scheme requires reconsideration.

Handling of the different media types: As the watermarking basis depends on the cover media (type), the question arouses, if embedding of higher alphabets is possible for all media types or if there are some types better suited? We have described one way for employing higher alphabets for audio just above. If other media types are qualified for similar methods remains in question.

Trade-off between code length and watermark complexity: Assuming watermark embedding allows the use of higher alphabets. It is still questionable that the shorter code lengths of the fingerprints, by means of higher alphabets, legitimates the corresponding increased complexity of the watermark embedding process?

Testing against new attacks: We must not forget that attack strategies also benefit from higher alphabets. Sophisticated (fingerprinting and watermarking) attacks that utilize the adjusted watermark embedding and detection have to be designed and evaluated. It is essential that new attacks do not render application of higher alphabets useless.

11.3 Sophisticated collaboration of watermarking and fingerprinting technologies

Fingerprinting codes are applied in watermarking scenarios in order to provide collusion security. To further increment the level of collusion security, in lieu of continuing research of fingerprinting codes alone, we suggest to focus on the interaction of the watermarking algorithm and the fingerprinting code.

Watermarking has continuously been optimized regarding the trade off between transparency and robustness (and capacity). By now – depending on the media type – watermarking achieves a level of transparency and robustness (and capacity) that is satisfactory for most use cases. Also, fingerprinting codes have continuously been optimized regarding optimality.

However, the respective optimization work proceeded independently from each other. Both technologies per se might not offer further space for optimization. But, in our mind, focusing on joint optimization work has the potential to pose new opportunities. That is, for instance, a watermarking embedding algorithm that offers collusion security by certain embedding tactics, or that is especially suited for fingerprint embedding. We already focused this idea in our approach published in [15]. There, we posted a watermarking embedding algorithm for audio files that uses properties of fingerprinting codes in order to provide collusion resistance. At the same time, the algorithm provides properties regarding robustness and transparency equal to the corresponding watermarking algorithm that is in economic use at Fraunhofer SIT. The downside of the approach is the increased complexity that refuses from wide application in practice.

11.4 Additional application scenarios for fingerprinting codes

Most research regarding fingerprinting codes was done against the background of media watermarking. Some other scenarios, see chapter 3.1, such as medical group testing or IPP codes, can also be found in literature. However the scenarios video game watermarking and hashtable databases, which are discussed in this thesis, were initiated by our research group and so far no other works are known. For a broad acceptance and application in practice, more work in this scenarios is required.

Fingerprinting for videogames:

Our research group initiated video game watermarking as a new application scenario of media watermarking [12], see chapter 3.6. We already published several watermarking algorithms for different media types and formats that are contained in video games, e.g. regarding watermarking 3D mesh models [118] or regarding DDS texture images [70]. Modifications of a watermarking algorithm in use at Fraunhofer SIT can be applied for the sound/audio files contained in games. However this field of research is only at its beginning. We find the general possibility to apply fingerprinting codes reasoned by the huge payload that is potentially available, but apart from our work, no one has broached the issue of collusion security in games yet. For the successful application of fingerprinting codes in video games more research is required in the following positions:

Sound watermarking algorithms: In this comparably new topic the embedding and detecting algorithms need adjustment for their especially use case. Some algorithms are first rough approaches and far from being optimal. For the application of integrity of fingerprinting codes in video games, the watermarking basis ought to be reliable and sound.

Fingerprinting codes for the individual media formats: In chapter 3.6 we stated to consider one single payload per game, in lieu of several payloads due to the different media types

contained in each game. To improve the security brought by fingerprinting codes, each media type – especially those that are essential for the game play – ought to obtain an individual and especially adjusted fingerprinting code.

Fingerprinting for databases:

Our approach presented in chapter 9 introduced fingerprinting codes as instrument of tracing back hashtable databases. To the best of our knowledge, this is still a unicum. Improvement can be achieved by adjusting proper fingerprinting codes to this scenario. However, the state of the art is limited to fingerprinting hashtable databases. Expansion to databases in general promises a wide application scenario.

Further research regarding fingerprinting for databases is necessary in the following positions:

Adjusting proper fingerprinting codes to this new scenario: As already mentioned in section 11.1, there might exist fingerprinting codes that with minor modifications offer appropriate properties and behavior. Also, we have to find out, if obsolete fingerprinting codes could prove beneficial in this new scenario.

Database properties: In order to apply fingerprinting codes for ordinary databases, the interaction of the properties of databases with the properties of fingerprinting codes need to be considered.

Attack scenario: New application scenarios always initiate the threat of new attacks. Before applying in practice, the potential attacks have to be considered. Resulting flaws in the fingerprinting codes need to be corrected.

Apart from that, we insist upon searching for further scenarios, which require collusion security and application of fingerprinting codes appears reasonably.

11.5 Combination of other security mechanisms and fingerprinting codes

In chapter 2 we already named other security mechanisms that can be combined with digital watermarking. Most of these mechanisms are also capable to be applied in combination with collusion secure fingerprinting. For instance, robust hashing is already in use for video game watermarking. There, the watermark (and fingerprint) information for one message is split into several parts and spread over a sequence of files of same and different format, see chapters 3.6 and 5.4. Hence, detection of the correct message requires correctly ordering the detected message parts which necessitates a technique ensuring that the corresponding files are recognized. As attackers can easily manipulate meta data of the files and thereby harden the recognition, a robust hash is less prone to manipulation and therefore is an appropriate technique to be combined with fingerprinting codes. More information is to be found in our publication for a robust 3D hash in [13].

Other security mechanisms, e.g. checksums and error correction codes are only limitedly applicable in combination with fingerprinting codes. Error correction codes are meant to resolve errors that can occur during transmission (or by manipulation). This means, the detected messages and the corresponding fingerprint is searched for errors. However, after a collusion attack, it is not clear in which direction the error needs to be corrected. As several watermark and fingerprint messages together are responsible for the detected message, an altering of the detected message by error correction codes not only does not provide any information regarding the attackers, but worse, will definitively falsify the manipulated fingerprint and thus harden tracing back.

Similar for checksums, which are meant to verify if the data has been altered. In a collusion scenario, several originators are responsible for the considered manipulated media copy. This automatically implies alteration of the media copy. However, in some scenarios, checksums could be supposedly applied together with fingerprinting codes. For example in case the watermarking system is constructed as such, that it embeds individual watermark messages for the fingerprint embedding and a fixed watermark message for other information, e.g. annotations or integrity watermarks as well. It is to find out, if there is a reasonable application scenario for this, or if the payload made available for the additional watermark information was better employed for longer and more secure fingerprinting codes.

Digital rights management (DRM) methods are not conflicting with digital watermarking. Therefore, application of fingerprinting codes in combination with DRM methods is possible.

We have to find out if there are any combinations of security mechanisms and fingerprinting codes reasonable and possible. To the best of our knowledge, a combination with most mechanisms is possible if watermarking per se is possible. Verification or confutation of this hypothesis is open for future research.

12 Summary and conclusions for future work

During the time of this thesis, the research of collusion secure fingerprinting codes gained an all-time high and many approaches – especially for information theoretic optimal codes – were published, many of those with a distinct impact on fingerprinting research, e.g. [86], [65], [55], [62], [64], [123], [61], [40], [53], [54], [75], to name a few selected ones. Within our work, we tried to involve the corresponding best approaches whenever possible and plausible, and we evaluated our optimizations with regards to those approaches as well. The lot of findings we attained are summed up in the following. Among the following four findings, the first two are completed by this research. However, the third and fourth findings motivate for some further research.

Discrepancy between theory and practice. Information theoretic optimal fingerprinting codes are based on unrealistic assumptions and parameter settings. Their propositions rarely hold true for application oriented fingerprinting with practical parameter settings. An asymptotically large value for the number of colluders is particularly inappropriate, because of its impact on the code length. Absolute collusion security independent of the collusion size is not possible. We classified the different application scenarios according to its probable need for collusion resistance, i.e. its maximum collusion size to be resistant against. The consequence is that we are not totally collusion secure but only to a distinct number of attackers, differing for every application scenario and proposed solution.

There is no *one fits it all* solution. Fingerprinting codes are application dependent, different applications demand different requirements and parameter settings. One solution that appears as best possible approach for a setting, is not necessary the best approach for a different setting. On the other hand this means, evaluating only a limited number of settings, the claims regarding optimality have to be stated with caution. We identified requirements for fingerprinting codes in several application scenarios with varying settings and proposed different solutions tailored to these requirements. So far no solutions able to satisfy all discussed scenarios have been found.

Practical fingerprinting codes still lack optimality. Probabilistic – asymptotically optimal – fingerprinting codes have been optimized to the maximum. With respect to an infinite number of colluders, optimal fingerprint generation algorithms and (recently) optimal tracing algorithms against the asymptotically best attack have been proposed. However, these approaches cannot prove – or have not proved so far – optimality with respect to e.g. practical limited collusion sizes. Neither did we prove optimality for those approaches. Instead for some approaches we proposed adjustments and modifications to ensure the targeted security level required for the respective application scenario. More efforts in further research are needed to also prove optimality for practical parameter settings.

No more significant improvements in fingerprinting research. Following the history of fingerprinting codes, we expect that large steps of improvement will not occur any more. Though much research has been done within the last three years, as already mentioned, its impact on practical settings, especially regarding code length, is rather low. Instead, significant improvements might be achieved in cooperation with the watermarking basis. As we exclusively focused on suitable fingerprinting codes for different application scenarios, we missed out the potential of a stronger cooperation of watermarking and fingerprinting in order to improve the possibilities of collusion security in practice. A focus on this cooperation appears promising for further research.

Within our examinations on the Tardos codes, we discovered that at several points within the constructions, modifications could transform a theoretical powerful fingerprinting scheme into a powerful scheme that can be applied in practice as well.

The developed algorithms are laid out for different application scenarios and different parameter requirements. The algorithms we presented in this thesis are as follows:

Zero false positive 2-secure fingerprinting: The approach differs from the other schemes presented as it ensures not to accuse any innocents and because it does not belong to the class of Tardos codes at all. Besides the guaranteed zero false positive error, the probability of not accusing any of the attackers is very low as well. Because of this and particularly because of the short code length provided – almost as short as ordinary transaction watermark messages – it can be applied for all application scenarios we identified and thus makes collusion security (against two attackers) practicable. The false negative error rate strongly depends on the selected Hamming Distance interval and the attack strategy and is therefore evaluated in empirical tests only. The strong statement of a zero false positive error is analytically proven and also confirmed in empirical tests (chapter 6 and appendix C).

Efficient and adaptable 3-secure fingerprinting: To be resistant against three attackers, we developed a simple structured fingerprinting scheme to be adjusted for the needs of appliers easily. Moreover, the approach provides two different tracing strategies to be selected depending on the (assumed) attack strategy and special tailored to it. Thereby we improved the performance and achieved an adaptable scheme demanding low computational effort compared to 3-secure approaches in literature. The manifold parameter settings prevent from giving a single analytical proof, therefore the soundness of the approach is stated by empirical tests (chapter 6.2 and appendix D).

Efficient fingerprinting (filter) tracing algorithm: The ranking search approach introduces a ranking into the columns of the fingerprint matrix according to significant positions in the manipulated fingerprint. Depending on the attack, the developed tracing algorithm first searches for positive disclosures of potential colluders according to the introduced ranking, before the (iterative) tracing algorithm applies a scoring function fed with the ranking information to trace (more) colluders. Besides the analytical soundness (chapter 7.1), the results are stated in empirical evaluations.

Accuracy optimizing fingerprint generation (correlative continuous I, II/discrete): Another step towards practical collusion security is achieved adjusting existing algorithms. We proposed three approaches regarding optimizing the fingerprint generation. The idea is to introduce correlations to the column generation, in order to reduce the number of outliers. These are by chance unfavorably generated fingerprints that significantly differ from the average. The correlated fingerprints result in a decreased standard deviation of the distribution of accusation scores. Applying a corresponding adjusted threshold this is capable to reduce the number of wrong accusations. We proposed the idea of the correlation first for a continuous distribution function, due to empirical observations. Based on this we developed a sophisticated approach employing stochastic measurements. Also we proposed the column correlation for a discrete distribution function. Analytical examinations that correspond to the proofs and considerations in [121] (appendix E and I) and empirical tests (chapters 7.2 and 8.2) confirm the improvements of the modifications in the fingerprint generation alone and in conjunctions with the corresponding tracing algorithms (see below).

Accuracy optimizing tracing algorithms adjusted to the optimized fingerprint generation:

Based on the results for the correlative fingerprint generation, we developed tracing algorithms adjusted to those modifications. Therefore we investigated adjusted decision thresholds special tailored to the corresponding generation process. The results are analytically sound as they correspond to the corresponding adaptations of the fingerprint generation (see above) and because they match the outcomes of the proofs of preliminary works, [121], [86] and [55]. This is also stated in large test sets and evaluations (chapters 7.4 and 8.3 as well as appendix G and J).

Attack and accusation score function independent tracing algorithm using mixture models:

Because of the recent publications of various tracing algorithms with optimal score functions in literature, we developed a tracing strategy that is independent of the collusion attack, the collusion size, and also (in particular) of the selected score function. We focused on the distribution of the scores and applied mixture models and expectation maximization strategies to best estimate the colluder-fingerprints. Thereby we are able to trace colluders more reliably especially in cases for which collusion strategy is unknown and the actual collusion size differs significantly from the expected collusion size. Results are presented for various accusation score functions and collusion sizes confirming the soundness of the approach (chapter 8.3 and appendix J).

Efficient and adaptable joint tracing algorithm: In cases for which computational complexity is negligible, joint tracing strategies are capable to improve the performance. That is, applying expectation maximization and log-likelihood scores for tuples of fingerprints result in decreased error rates. Pruning fingerprints in the first step, our approach also reduces the computational complexity compared to approaches in literature. Different test sets (chapter 8.1 and appendix H) confirm the analytical results that correspond to the preliminary approach of [74].

Fingerprinting codes for database tracing: Apart from the media watermarking application scenarios, we also proposed solutions to employ collusion security for (hashtable) databases. Blending dummy hashes into the hash tables of databases, we are able to individualize hash tables for each accessor. The individual dummy hashes represent the fingerprint positions clearly identifying the corresponding accessor. We investigated different scenario dependent attacks and tested them against our approach to prove its functionality. We also tested possible collusion attacks from the classical media fingerprinting scenario.

We developed a series of innovative collusion secure fingerprinting codes to be applied in conjunction with digital watermarking. Our focus is on practicability and security in order to suit the varying application scenarios we identified. Most approaches are analytically proven and all are empirically evaluated.

If applied with the help of continuative security mechanisms, first and foremost with digital transaction watermarking, the proposed fingerprinting codes satisfy the requirements we identified for the respective application scenarios. The different scenarios are supported with fingerprinting codes specially tailored for the respective requirements, thereby building the bridge between watermarking practice and theory of collusion secure codes. The results are summarized in chapter 10.

Though we accomplished many challenges within our work on fingerprinting codes, some challenges remain open. In some application scenarios the challenge of collusion resistance against

large collusions could not be solved. That is, application scenarios naturally providing very limited watermarking payload for which the required high security level could not be ensured.

Besides, we acted on the strong assumption of a perfect watermarking basis, i.e. the watermark message always is detected correctly. This assumption obviously does not mirror practical settings. On the one hand, attackers can also perform media processing attacks in conjunction with collusion attacks, thereby possibly inducing detection (bit) errors. The respective positions in the manipulated fingerprint contains fraudulent information that tentatively raises the error rates. On the other hand, comparing their media copies, attackers cannot receive fingerprint symbols, but differing media information only. Consequently, the number of detectable positions for the collusion attack in most cases is smaller thus weakening their attack. Hence, for application in practice the proposed solutions ought to be applied with respect to this imbalance. In chapter 11 we listed further open challenges as well as corresponding concepts and ideas.

References

- [1] Iso/iec 18004:2006 information technology – automatic identification and data capture techniques – qr code 2005 bar code symbology specification.
- [2] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover, 1965.
- [3] R. Agrawal and J. Kiernan. Watermarking relational databases. In *In 28th Int. Conference on Very Large Databases, Hong Kong, 2002*.
- [4] M. Al-Hames and G. Rigoll. Ein Treffen mit mehreren Gauss und wie der EM-Algorithmus sogar denen noch etwas beibringt, 2006.
- [5] E. Allamanche. Content-based Identification of Audio Material Using MPEG7 Low Level Description. In *International Symposium/Conference on Music Information Retrieval, 2001*.
- [6] N. Alon, E. Fischer, and M. Szegedy. Parent-identifying codes. *Journal of Combinatorial Theory, Series A*, 95(2), 2001.
- [7] E. Amiri and G. Tardos. High rate fingerprinting codes and the fingerprinting capacity. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '09, Philadelphia, PA, USA, 2009*. Society for Industrial and Applied Mathematics.
- [8] N. P. Anthapadmanabhan, A. Barg, and I. Dumer. Fingerprinting capacity under the marking assumption. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, june 2007.
- [9] W. Berchtold. Optimierung der Robustheit und Klangqualität digitaler Audio-Wasserzeichen-Verfahren im Kontext von angepassten Vorwärtsfehlerkorrektur-Algorithmen. Master's thesis, University of Applied Science Darmstadt, December 2008.
- [10] W. Berchtold and M. Schäfer. Performance and code length optimization of joint decoding Tardos fingerprinting. In *Proceedings of the on Multimedia and security, MM&Sec '12, New York, NY, USA, 2012*. ACM.
- [11] W. Berchtold and M. Schäfer. Rebound on Symmetric Tardos Codes. In *Intelligent Information Hiding and Multimedia Signal Processing, 2012. IHHMSP '12 International Conference on*, 2012.
- [12] W. Berchtold, M. Schäfer, H. Liu, F. Touceira Takahashi, A. Schmitz, S. Zmudzinski, M. Steinebach, and J. Wieneke. Video Game Watermarking. *IS&T SPIE Electronic Imaging*, 2013.
- [13] W. Berchtold, M. Schäfer, M. Rettig, and M. Steinebach. Robust hashing for 3d models. In *Proceedings of Electronic Imaging 2014 - Media Watermarking, Security, and Forensics 2014*, Proceedings of SPIE. IS&T/SPIE, Feb 2014.
- [14] W. Berchtold, M. Schäfer, and M. Steinebach. Leakage detection and tracing for databases. In *Proceedings of the first ACM workshop on Information Hiding and Multimedia Security (IH&MMSec 2013), June 17-19, 2013 Montpellier, France, IH&MMSec '13, New York, NY, USA, June 2013*. ACM.

-
- [15] W. Berchtold, S. Zmudzinski, M. Schäfer, and M. Steinebach. Collusion-secure patchwork embedding for transaction watermarking. In *Proceeding of Electronic Imaging 2011 - Media Watermarking, Security, and Forensics XIII*. IS&T SPIE, Jan 2011.
- [16] O. Berkman, M. Parnas, and J. Sgall. Efficient dynamic traitor tracing. *SIAM Journal on Computing*, 30(6), 2001.
- [17] E. Biham and O. Dunkelman. A framework for iterative hash functions. *ePrint report 2007/278 HAIFA*, 2007.
- [18] J. Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, International Computer Science Institute Berkeley CA, and Computer Science Division Department of Electrical Engineering and Computer Science U.C. Berkeley, 1998.
- [19] S. Blackburn. Frameproof codes. *SIAM Journal on Discrete Mathematics*, 16(3), 2003.
- [20] G. R. Blakley, C. Meadows, and G. B. Purdy. Fingerprinting long forgiving messages. In H. Williams, editor, *Advances in Cryptology, CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 1986.
- [21] O. Blayer and T. Tassa. Improved versions of Tardos' fingerprinting scheme. *Designs, Codes and Cryptography*, 48, 2008. 10.1007/s10623-008-9200-z.
- [22] D. Boesten and B. Škorić. Asymptotic fingerprinting capacity in the combined digit model. In M. Kirchner and D. Ghosal, editors, *Information Hiding*, volume 7692 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013.
- [23] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5), Sept. 1998.
- [24] L. Boney, A. H. Tewfik, and K. N. Hamdy. Digital watermarks for audio signals. In *International Conference on Multimedia Computing and Systems*, 1996.
- [25] J. Buchmann. *Einführung in die Kryptographie*. Springer-Lehrbuch. Springer, 2008.
- [26] J. Cotrina-Navau, M. Fernandez, and M. Soriano. A family of collusion 2-secure codes. In *Information Hiding*. Springer Berlin / Heidelberg, 2005.
- [27] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoan. Secure spread spectrum watermarking for multimedia. *Image Processing, IEEE Transactions on*, 6(12), 1997.
- [28] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoan. Secure spread spectrum watermarking for images, audio and video. In *INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, PROCEEDINGS - VOL III.*, 1996.
- [29] I. J. Cox, M. L. Miller, and J. A. Bloom. Watermarking applications and their properties. In *Information Technology: Coding and Computing, 2000. Proceedings. International Conference on*, 2000.
- [30] I. J. Cox, M. L. Miller, and J. A. Bloom. *Digital Watermarking*. Morgan Kaufmann, 2002.
- [31] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography (Second Edition)*. Elsevier Science Publishers B. V., 2007.
- [32] J. Dittmann. *Digitale Wasserzeichen: Grundlagen, Verfahren, Anwendungsgebiete*. Springer, auflage: 2000 edition, April 2000.

-
- [33] J. Dittmann, M. Steinebach, I. Rimac, S. Fischer, and R. Steinmetz. Combined video and audio watermarking: Embedding content information in multimedia data, 2000.
- [34] J. Dittmann, P. Wohlmacher, and K. Nahrstedt. Using cryptographic and watermarking algorithms. *MultiMedia, IEEE*, 8(4), 2001.
- [35] A. Duel-Hallen. Decorrelating decision-feedback multiuser detector for synchronous code-division multiple-access channel. *Communications, IEEE Transactions on*, 41(2), 1993.
- [36] M. Fenzi, H. Liu, M. Steinebach, and R. Caldelli. Markov random fields pre-warping to prevent collusion in image transaction watermarking. In *Proceeding of the Seventh IASTED International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA 2010), Innsbruck, Austria*, 2010.
- [37] A. Fiat and T. Tassa. Dynamic traitor tracing. In *Advances in Cryptology-CRYPTO'99*. Springer, 1999.
- [38] J. Fridrich and M. Goljan. Robust hash functions for digital watermarking. In *Information Technology: Coding and Computing, 2000. Proceedings. International Conference on*, 2000.
- [39] S. Fujitsu, K. Nuida, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, and H. Imai. A tracing algorithm for short 2-secure probabilistic fingerprinting codes strongly protecting innocent users. In *Proc. 4th IEEE Consumer Communications and Networking Conference CCNC 2007*, 11–13 Jan. 2007.
- [40] T. Furon, A. Guyader, and F. C erou. Decoding Fingerprinting Using the Markov Chain Monte Carlo Method. In *WIFS - IEEE Workshop on Information Forensics and Security*, Tenerife, Spain, Dec. 2012. IEEE.
- [41] T. Furon and L. Perez-Freire. Worst case attacks against binary probabilistic traitor tracing codes. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, dec. 2009.
- [42] E. G omez, P. Cano, L. Gomes, E. Batlle, and M. Bonnet. Mixed watermarking-fingerprinting approach for integrity verification of audio recordings. In *International Telecommunications Symposium, ITS2002, Natal, Brazil*, 2002.
- [43] M. Hagiwara, G. Hanaoka, and H. Imai. A short random fingerprinting code against a small number of pirates. In M. Fossorier, H. Imai, S. Lin, and A. Poli, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, volume 3857 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006.
- [44] M. Hagiwara, T. Yoshida, and H. Imai. Bounds on the number of users for random 2-secure codes. In *AAECC*, 2009.
- [45] J. Haitzma and T. Kalker. A watermarking scheme for digital cinema. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 2, Oct 2001.
- [46] F. H. Hartung, J. K. Su, and B. Girod. Spread spectrum watermarking: malicious attacks and counterattacks, 1999.
- [47] J. F. Healey. *Statistics: A Tool for Social Research*. CENGAGE Learning, 2014.
- [48] H. D. Hollmann, J. H. Van Lint, J.-P. Linnartz, and L. M. Tolhuizen. On codes with the identifiable parent property. *Journal of Combinatorial Theory, Series A*, 82(2), 1998.

-
- [49] H. D. L. Hollmann, J. H. van Lint, J.-P. Linnartz, L. M. G. M. Tolhuizen, and H. D. Hollmann. On codes with the identifiable parent property, 1998.
- [50] Y.-W. Huang and P. Moulin. Capacity-achieving fingerprint decoding. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, dec. 2009.
- [51] Y.-W. Huang and P. Moulin. Saddle-point solution of the fingerprinting capacity game under the marking assumption. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, 28 2009-july 3 2009.
- [52] Y.-W. Huang and P. Moulin. On the saddle-point solution and the large-coalition behavior of fingerprinting games. *CoRR*, abs/1011.1261, 2010.
- [53] Y.-W. Huang and P. Moulin. On fingerprinting capacity games for arbitrary alphabets and their asymptotics. In *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, July 2012.
- [54] Y.-W. Huang and P. Moulin. On the saddle-point solution and the large-coalition asymptotics of fingerprinting games. *IEEE Transactions on Information Forensics and Security*, 7(1), 2012.
- [55] S. Ibrahim, B. Škorić, and J.-J. Oosterwijk. Riding the saddle point: asymptotics of the capacity-achieving simple decoder for bias-based traitor tracing. *EURASIP Journal on Information Security*, 2014(1), 2014.
- [56] N. F. Johnson, Z. Duric, and S. Jajodia. *Information Hiding: Stenography and Watermarking - Attacks and Countermeasures*. Kluwer Academic Publishers, 2000.
- [57] T. Kalker, G. Depovere, J. Haitsma, and M. Maes. Video watermarking system for broadcast monitoring. volume 3657. IS&T SPIE, 1999.
- [58] A. Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, vol. IX, January 1883.
- [59] T. Kitagawa, M. Hagiwara, K. Nuida, H. Watanabe, and H. Imai. A group testing based deterministic tracing algorithm for a short random fingerprint code. In *Proc. International Symposium on Information Theory and Its Applications ISITA 2008*, 7–10 Dec. 2008.
- [60] W. R. Knight. A Computer Method for Calculating Kendall's Tau with Ungrouped Data. *Journal of the American Statistical Association*, 61(314), June 1966.
- [61] M. Kuribayashi. Bias equalizer for binary probabilistic fingerprinting codes. In *Proceedings of the 14th International Conference on Information Hiding, IH'12, Berlin, Heidelberg, 2013*. Springer-Verlag.
- [62] T. Laarhoven. Capacities and capacity-achieving decoders for various fingerprinting games. *CoRR*, abs/1401.5688, 2014.
- [63] T. Laarhoven, J. Doumen, P. Roelse, B. Škorić, and B. de Weger. Dynamic tardos traitor tracing schemes. *Information Theory, IEEE Transactions on*, 59(7), 2013.
- [64] T. Laarhoven and B. Weger. Optimal symmetric tardos traitor tracing schemes. *Designs, Codes and Cryptography*, 71(1), 2014.
- [65] T. Laarhoven and B. d. Weger. Discrete distributions in the tardos scheme, revisited. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec '13, New York, NY, USA, 2013*. ACM.

-
- [66] T. M. M. Laarhoven. Collusion-resistant traitor tracing schemes. Master's thesis, Eindhoven : Technische Universiteit Eindhoven, 2011.
- [67] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM J. on Optimization*, 9(1), May 1998.
- [68] Y. Li and R. H. Deng. Publicly verifiable ownership protection for relational databases. *ASIACCS*, 2006.
- [69] S. Lindsey, R. Hertwig, and G. Gigerenzer. Communicating statistical dna evidence. *Jurimetrics*, 2003.
- [70] H. Liu, W. Berchtold, M. Schäfer, and M. Steinebach. Texture watermarking for video games. In *Proceedings of Electronic Imaging 2014 - Media Watermarking, Security, and Forensics 2014*, Proceedings of SPIE. IS&T/SPIE, Feb 2014.
- [71] S. Mair. Bachelor Thesis: "Optimierung eines koalitionsresistenten Fingerprinting Codes", Fraunhofer SIT & University of Applied Sciences Darmstadt, 2013.
- [72] P. Meerwald and T. Furon. Group testing meets traitor tracing. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011.
- [73] P. Meerwald and T. Furon. Iterative single tardos decoder with controlled probability of false positive. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, july 2011.
- [74] P. Meerwald and T. Furon. Towards joint tardos decoding: The 'don quixote' algorithm. In *Information Hiding - 13th International Conference, IH 2011, Prague, Czech Republic, May 18-20, 2011, Revised Selected Papers*, 2011.
- [75] P. Meerwald and T. Furon. Towards practical joint decoding of binary Tardos fingerprinting codes. *IEEE Transactions on Information Forensics and Security*, 7(4), Apr. 2012.
- [76] P. Moulin. Universal fingerprinting: Capacity and random-coding exponents. *CoRR*, abs/0801.3837, 2008.
- [77] J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 231(694-706), 1933.
- [78] P. Nguyen. State Of The Media - The Cross Platform Report Q2 2012. Technical report, Nielsen, 2012.
- [79] K. Nuida. An error-tolerant variant of a short 2-secure fingerprint code and its security evaluation. In *IWSEC*, 2009.
- [80] K. Nuida. An improvement of short 2-secure fingerprint codes strongly avoiding false-positive. In S. Katzenbeisser and A.-R. Sadeghi, editors, *Information Hiding*, volume 5806 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009.
- [81] K. Nuida. A general conversion method of fingerprint codes to (more) robust fingerprint codes against bit erasure. In *Proceedings of the 4th international conference on Information theoretic security, ICITS'09*, Berlin, Heidelberg, 2010. Springer-Verlag.

-
- [82] K. Nuida. Short collusion-secure fingerprint codes against three pirates. In *Proceedings of the 12th international conference on Information hiding, IH'10*, Berlin, Heidelberg, 2010. Springer-Verlag.
- [83] K. Nuida, S. Fujitsu, M. Hagiwara, H. Imai, T. Kitagawa, K. Ogawa, and H. Watanabe. An Efficient 2Secure and Short Random Fingerprint Code and Its Security Evaluation. *Ieice Transactions*, 92-A, 2009.
- [84] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, and H. Imai. An improvement of discrete tardos fingerprinting codes. *Des. Codes Cryptography*, 52, September 2009.
- [85] K. Nuida, M. Hagiwara, H. Watanabe, and H. Imai. Optimization of tardos's fingerprinting codes in a viewpoint of memory amount. In T. Furon, F. Cayre, G. Doërr, and P. Bas, editors, *Information Hiding*, volume 4567 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2007.
- [86] J.-J. Oosterwijk, B. Škorić, and J. Doumen. Optimal suspicion functions for tardos traitor tracing schemes. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec '13*, New York, NY, USA, 2013. ACM.
- [87] L. Perez-Freire and T. Furon. Blind decoder for binary probabilistic traitor tracing codes. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, 2009.
- [88] N. Reimers. Optimization of Binary Probabilistic Fingerprinting Codes for Real World Watermarking. Master's thesis, Technical University of Darmstadt and Fraunhofer Institute for Secure Information Technologies SIT, 2014.
- [89] L. Sachs. *Angewandte Statistik - Anwendungen statistischer Methoden*. Springer Verlag Berlin, Heidelberg, New York, 1999.
- [90] P. Sarkar and D. R. Stinson. Frameproof and ipp codes. In C. Rangan and C. Ding, editors, *Progress in Cryptology INDOCRYPT 2001*, volume 2247 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2001.
- [91] M. Schäfer, W. Berchtold, M. Heilmann, S. Zmudzinski, M. Steinebach, and S. Katzenbeisser. Collusion secure fingerprint watermarking for real world applications. In *Proc. of GI-Sicherheit 2010*. Gesellschaft für Informatik, Okt 2010.
- [92] M. Schäfer, W. Berchtold, and M. Steinebach. Fast and adaptive tracing strategies for 3-secure fingerprint watermarking codes. In *Proceedings of the 11th annual ACM workshop on Digital rights management, DRM '11*, New York, NY, USA, 2011. ACM.
- [93] M. Schäfer, W. Berchtold, and M. Steinebach. Ranking search for probabilistic fingerprinting codes. In *Proceedings of Electronic Imaging 2012 - Media Watermarking, Security, and Forensics 2012*, Proceedings of SPIE. IS&T/SPIE, Jan 2012.
- [94] M. Schäfer, W. Berchtold, S. Zmudzinski, and M. Steinebach. Zero false positive 2-secure fingerprinting watermarking based on combining hamming distance conditions and parent pair search. In *Proceedings of the 12th ACM workshop on Multimedia and security, MMSec '10*, New York, NY, USA, 2010. ACM.
- [95] M. Schäfer, W. Berchtold, S. Zmudzinski, and M. Steinebach. Verfahren zur Erzeugung von Transaktionswasserzeichen und Auswerteverfahren zur Kundenrückverfolgung, 2012. DE Patent App. DE201,010,044,228.

-
- [96] M. Schäfer, S. Mair, W. Berchtold, and M. Steinebach. Universal Threshold Calculation for Fingerprinting Decoders using Mixture Models. In *Proceedings of the 3rd ACM Workshop on Information Hiding and Multimedia Security, IH&MMSec 2015, Portland, OR, USA, June 17 - 19, 2015*, 2015.
- [97] B. Schneier. *Angewandte Kryptographie . Protokolle, Algorithmen und Sourcecode in C (Informationssicherheit)*. Angewandte Kryptographie . Protokolle, Algorithmen und Sourcecode in C (Informationssicherheit), 5 edition, May 1996.
- [98] B. Schneier and C. Hall. An improved e-mail security protocol. *Computer Security Applications Conference*, 1997.
- [99] F. Sebe and J. Domingo-Ferrer. Short 3-secure fingerprinting codes for copyright protection. In *7th Australasian Conference on Information Security and Privacy*, 2002.
- [100] J. Seitz. *Digital Watermarking for Digital Media*. IGI Publishing, May 2005.
- [101] H. N. Shaikh. Bachelor Thesis: "Intergration of Fingerprinting Codes for Image Watermarking Application", Fraunhofer SIT & Faculty of Engineering and Computer Science of the University of Applied Sciences Hamburg, 2013.
- [102] A. Simone and B. Skoric. Accusation probabilities in tardos codes: beyond the gaussian approximation. *Des. Codes Cryptography*, 63(3), 2012.
- [103] A. Simone and B. Škorić. False negative probabilities in tardos codes. *Designs, Codes and Cryptography*, 74(1), 2015.
- [104] R. Sion. Proving ownership over categorical data. *IEEE International Conference on Data Engineering*, 2004.
- [105] M. Steinebach. *Digitale Wasserzeichen für Audiodaten*. PhD thesis, Fraunhofer - Informations- und Kommunikationstechnik - Gruppe, 2004.
- [106] M. Steinebach and J. Dittmann. Design principles for active audio and video fingerprinting. *Multimedia Technologies: Concepts, Methodologies, Tools, and Applications*, 2005. Lu, Chun-Shien (Verf.): Multimedia security : stenography and digital watermarking techniques for protection of intellectual property. Hershey, Pa. : Idea, 2005, S. 157 - 172.
- [107] M. Steinebach, E. Hauer, and P. Wolf. Efficient watermarking strategies. In *Proc. Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution AXMEDIS '07*, 28–30 Nov. 2007.
- [108] M. Steinebach, H. Liu, and Y. Yannikos. Forbild: Efficient robust image hashing. In SPIE, editor, *SPIE-IS&T Electronic Imaging, Media Watermarking, Security, and Forensics 2012*, SPIE volume 8303, 830300, January 2012, 2012.
- [109] M. Steinebach and S. Zmudzinski. Countermeasure for collusion attacks against digital watermarking. In *Electronic Imaging 2006*, volume 6072. International Society for Optics and Photonics, 2006.
- [110] M. Steinebach and S. Zmudzinski. Evaluation of robustness and transparency of multiple audio watermark embedding. In I. Delp, Edward J., P. W. Wong, J. Dittmann, and N. D. Memon, editors, *Proceeding of SPIE Int., Symposium on Electronic Imaging, Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, San Jose, USA*, volume 6819, 2008.

-
- [111] R. Steinmetz. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer Verlag, Oct 2000. 3. Auflage (erstmalig mit CD).
- [112] D. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM Journal on Discrete Mathematics*, 11(1), 1998.
- [113] A. S. Tanenbaum. *Computer Networks*. Pearson Education, 4 edition, August 2002.
- [114] G. Tardos. Optimal probabilistic fingerprinting codes. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 116-125, 2003.
- [115] T. Tassa. Low bandwidth dynamic traitor tracing schemes. *Journal of cryptology*, 18(2), 2005.
- [116] S. Thiemert. *Digitale Wasserzeichen zum Integritätsschutz von Videodaten*. PhD thesis, Technical University of Darmstadt, 2013.
- [117] D. Trick. Digital Watermarking for 3D-Models in Video Games. Master's thesis, Technical University of Darmstadt, June 2013.
- [118] D. Trick, W. Berchtold, M. Schäfer, and M. Steinebach. 3D Watermarking in the Context of Video Games. In *Proceeding IEEE 15th International Workshop on Multimedia Signal Processing 2013 (MMSP 2013), Pula, Italy, September 30 - October 2, 2013*. IEEE, Oct. 2013.
- [119] T. Trung and S. Martirosyan. New constructions for ipp codes. *Designs, Codes and Cryptography*, 35(2), 2005.
- [120] M. Varanasi and T. Guess. Optimum decision feedback multiuser equalization with successive decoding achieves the total capacity of the gaussian multiple-access channel. In *Signals, Systems amp; Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on*, volume 2, 1997.
- [121] B. Škorić, S. Katzenbeisser, and M. Celik. Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. *Designs, Codes and Cryptography*, 46, 2008.
- [122] B. Škorić, S. Katzenbeisser, H. Schaathun, and M. Celik. Tardos fingerprinting codes in the combined digit model. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, Dec 2009.
- [123] B. Škorić and J.-J. Oosterwijk. Binary and q-ary tardos codes, revisited. *Designs, Codes and Cryptography*, 2013.
- [124] B. Škorić, T. U. Vladimirova, M. Celik, and J. C. Talstra. Tardos Fingerprinting is Better Than We Thought. *Information Theory, IEEE Transactions on*, 54(8), Aug. 2008.
- [125] N. R. Wagner. Fingerprinting. In *Proceedings of the 1983 IEEE Symposium on Security and Privacy, SP '83, Washington, DC, USA, 1983*. IEEE Computer Society.
- [126] P. Wolf, E. Hauer, and M. Steinebach. The video watermarking container - efficient real-time transaction watermarking. In *SPIE Proceedings of Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, vol. 6819*, 2009.
- [127] A. Wyner. Shannon-theoretic approach to a gaussian cellular multiple-access channel. *Information Theory, IEEE Transactions on*, 40(6), 1994.

The appendix contains additional discussions, proofs and tests regarding the different approaches proposed in chapters 6 through 9. In addition we take on the explanations and discussions about the fingerprint generation process and the stateless attacks from chapter 4.

A Optimal parameter selection for the symmetric Tardos scheme

The symmetric *Tardos* scheme described in chapter 4.5 requires the selection of the code length $m = d_m c_0^2 \ln(1/\varepsilon_1)$ and the cutoff $t = 1/(d_t c_0)$ for the fingerprint generation process. The corresponding tracing algorithm requires a decision threshold $Z = d_z c_0 \ln(1/\varepsilon_1)$. This section upgrades the selection of these parameters for the symmetric *Tardos* scheme according to the approach recently published in [64].

Parameter Selection for the Continuous Bias Distribution

Tardos presented in his original scheme some fixed choices for the different parameters: Code length $m = 100c_0^2 \lceil \ln(1/\varepsilon_1) \rceil$, threshold $Z = 20c_0 \lceil \ln(1/\varepsilon_1) \rceil$, and cutoff $t = 1/(300c_0)$ and proved ε_1 -soundness and (c_0, ε_2) -completeness for $\varepsilon_2 = \varepsilon_1^{c_0/4}$.

For a practical usage of collusion secure fingerprints, outputting innocent fingerprints (false positive) is generally seen as worse than not detecting any colluder fingerprint at all (false negative). The choice $\varepsilon_2 = \varepsilon_1^{c_0/4} \ll \varepsilon_1$ for $c_0 > 4$ is therefore quite unfavorable. Increasing ε_2 , while keeping ε_1 fixed, the code length can be reduced [124]. Instead of this fixed ratio between ε_1 and ε_2 we consider both error rates uncoupled. A quite common selection for ε_2 is $\varepsilon_2 = 1/2$, while ε_1 is chosen comparably small, e.g. $\varepsilon_1 = 10^{-6}$.

In [64] asymptotically tight security proofs for the arcsine-distribution and the symmetric score function are published. These show that asymptotically it holds $d_m \rightarrow \pi^2/2$. It is also presented a prescription for the selection of d_z and d_t in order to minimize d_m for concrete values of c_0 , ε_1 and ε_2 . For the derived values, ε_1 -soundness and (c_0, ε_2) -completeness is proven.

In chapter 4.3 we already introduced the notation $\eta = \log(\varepsilon_2)/\log(\varepsilon_1)$ such that $\varepsilon_2 = \varepsilon_1^\eta$. This allows to describe the size of ε_2 with respect to ε_1 .

In [64] was proven ε_1 -soundness and (c_0, ε_2) -completeness for the case $c_0 \geq 2$ and $\varepsilon_2 \geq \varepsilon_1$ (i.e. $\eta \leq 1$) for the following set of parameters:

$$d_m = 23.79, \quad d_z = 8.06, \quad d_t = 28.31.$$

In the asymptotic case of $c_0 \rightarrow \infty$, the asymptotically tight parameters for ε_1 -soundness and (c_0, ε_2) -completeness are:

$$m \rightarrow \frac{\pi^2}{2} c_0^2 \ln(1/\varepsilon_1), \quad Z \rightarrow \pi c_0 \ln(1/\varepsilon_1), \quad t \rightarrow \frac{\gamma}{4} c_0^{-4/3}$$

with $\gamma = (\frac{2}{3\pi})^{2/3}$.

It was also presented the following equations that lead to a minimization of d_m with ε_1 -soundness and (c_0, ε_2) -completeness.

Theorem 2 Let $\eta = \log(\varepsilon_2)/\log(\varepsilon_1)$, c_0 be given, and let r, s, g be fixed, satisfying $r \in (1/2, \infty)$, $s \in (0, \infty)$, $g \in (0, 2/\pi)$. Define the function $h^{-1} : (0, \infty) \rightarrow (1/2, \infty)$, $h^{-1}(x) = (e^x - 1 - x)/x^2$ while the function $h : (1/2, \infty) \rightarrow (0, \infty)$ denotes its inverse function so that $e^x \leq 1 + x + \lambda x^2$ for all $x \leq h(\lambda)$.

We define the following four conditions:

$$d_\alpha \geq \frac{\sqrt{d_t}}{h(r)\sqrt{c_0}} \quad (S1)$$

$$\frac{d_z}{d_\alpha} - \frac{r d_m}{d_\alpha^2} \geq 1 \quad (S2)$$

$$\frac{2 - \frac{4}{d_t}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_t}c_0} \geq g \quad (C1')$$

$$g d_m - d_z \geq \eta \sqrt{\frac{d_t}{s^2 c_0}}. \quad (C2)$$

Then the optimal choice of d_t, d_α , and d_z that minimizes d_m and satisfying conditions (S1), (S2), (C1'), (C2), is given by:

$$\begin{aligned} d_t &= \left(\frac{1}{\frac{4}{\pi} - 2g} \left(\sqrt{\frac{(h^{-1}(s)s)^2}{c_0} + \frac{16}{\pi} \left(\frac{2}{\pi} - g \right)} + \frac{h^{-1}(s)s}{\sqrt{c}} \right) \right)^2 \\ d_\alpha &= \max \left(\frac{\sqrt{d_t}}{h(r)\sqrt{c_0}}, \frac{r}{g} + \sqrt{\left(\frac{r}{g} \right)^2 + \frac{r}{g} \eta \sqrt{\frac{d_t}{s^2 c_0}}} \right) \\ d_z &= \frac{g d_\alpha^2 + r \eta \sqrt{\frac{d_t}{s^2 c_0}}}{g d_\alpha - r} \\ d_m &= \frac{\eta \sqrt{\frac{d_t}{s^2 c_0}} + d_z}{g} \end{aligned} \quad (O4)$$

It was shown, that if the inequalities (S1), (S2), (C1'), (C2) are fulfilled, the scheme with the corresponding parameters d_m, d_z, d_t is ε_1 -sound and (c_0, ε_2) -complete. In order to minimize the code length for given η and given c_0 , one has to find the triple (r, s, g) with $r \in (1/2, \infty)$, $s \in (0, \infty)$, $g \in (0, 2/\pi)$ that minimizes the right hand side of (O4) such that (S1), (S2), (C1'), (C2) are fulfilled. This minimization problem can be solved by the Nelder-Mead simplex method [67].

Figure 17 shows that the code length parameter d_m converges fairly slowly towards $\pi^2/2 \approx 4.93$. In practical scenarios, where c_0 is comparably small, the code length parameter d_m is larger than $\pi^2/2$ for provable security. Table 21 depicts the computed values for $c_0 \leq 8$. If the system has 10.000 fingerprints and the probability that any innocent fingerprint is output is required to be below $1/1000$, ε_1 has to be selected as $\varepsilon_1 = 10^{-8}$. A secure fingerprinting scheme that resists up to five colluders with a probability of a false negative error of $\varepsilon_2 = 1/2$ requires a code length of at least $m = 9.23 \cdot 5^2 \cdot \ln(1/10^{-8}) \approx 4250.57$ bits.

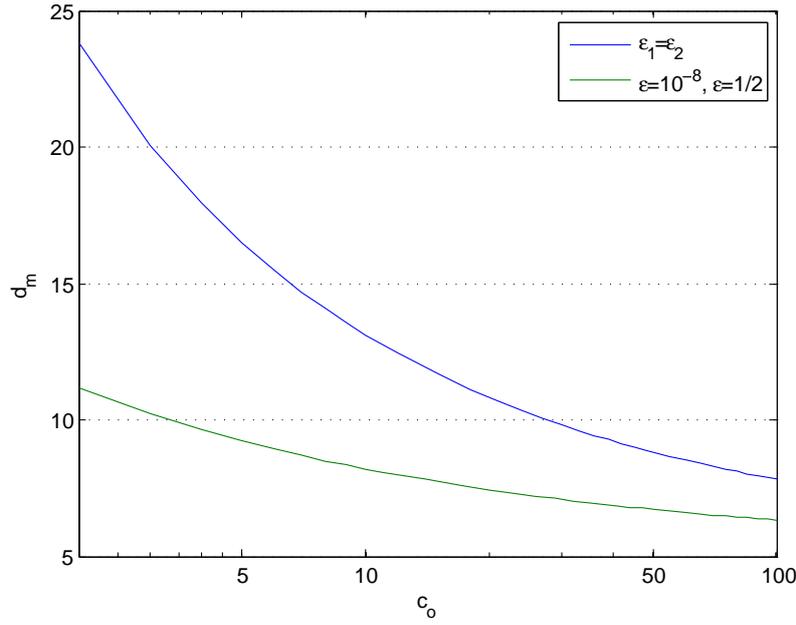


Figure 17: Minimal code length parameter d_m at different collision sizes for given error bounds $\varepsilon_1 = \varepsilon_2$ (upper line) and $\varepsilon_1 = 10^{-8}, \varepsilon_2 = 1/2$ (bottom line). Based on [88]

Table 21: Parameters d_m, d_z, d_t for minimal code length for up to 8 colluder.

		c_0						
		2	3	4	5	6	7	8
$\varepsilon_1 = \varepsilon_2$	d_m	23.79	20.06	17.93	16.52	15.49	14.70	14.08
	d_z	8.06	7.30	6.84	6.52	6.28	6.09	5.93
	d_t	28.30	28.93	29.56	30.15	30.71	31.23	31.72
$\varepsilon_1 = 10^{-8}, \varepsilon_2 = 1/2$	d_m	11.16	10.21	9.63	9.23	8.93	8.70	8.51
	d_z	5.83	5.47	5.23	5.07	4.94	4.85	4.77
	d_t	26.65	28.47	29.94	31.20	32.31	33.31	34.22

Integral Code Lengths

The code length for the symmetric *Tardos* scheme is parameterized as $m = d_m c_0^2 \ln(1/\varepsilon_1)$. As one might have noticed, m is not necessarily an integer as neither d_m nor $\ln(1/\varepsilon_1)$ must be an integer. But in practice an integral code length is required. *Tardos* solved this by setting the code length to $m = 100c_0^2 \lceil \ln(1/\varepsilon_1) \rceil$ [114]. But rounding up d_m and $\ln(1/\varepsilon_1)$ might increase the code length significantly. With the example above, but instead using $m = \lceil 9.23 \rceil \cdot 5^2 \cdot \lceil \ln(1/10^{-8}) \rceil = 4750$ bits results to an 11.8% longer code. Hence, instead of rounding up the different factors, rounding up the entire term would be a better choice: $m = \lceil d_m c_0^2 \ln(1/\varepsilon_1) \rceil$. In that case the code length increases at most 1 bit. However, the different security proofs in literature usually use the code length $m = d_m c_0^2 \ln(1/\varepsilon_1)$. The security proofs for soundness and completeness may not work any longer for $m' = \lceil m \rceil$. But *Laarhoven* and *de Weger* prove in [64], that if $(d_m, d_z, d_t, d_\alpha, r, s, g)$ is a septuple satisfying conditions (S1), (S2), (C1'), (C2) giving the scheme

parameters $m = d_m c_0^2 \ln(1/\varepsilon_1)$, $Z = d_z c_0 \ln(1/\varepsilon_1)$ and $t = 1/(d_t c_0)$, then the scheme with parameters

$$m' = \lceil m \rceil, \quad Z' = Z + \frac{g}{c_0} (\lceil m \rceil - m), \quad t' = t$$

is also ε_1 -sound and (c_0, ε_2) -complete.

B Regarding the stateful attack strategies

This section expands the discussion about sophisticated collusion attacks on fingerprinting codes that was initiated in chapter 4.6 and shows another stateful attack strategy.

As mentioned in chapter 4.6, a typical assumption in literature is the location independence of the attack strategy. Due to definition 20, the *stateless* attack model does not depend on the index i nor on the symbols for the other detectable positions or the selected output for other positions. This section extends the discussion about these type of attacks. It explains the *Greedy attack* of chapter 4.6 in more detail and presents another attack belonging to this class, that is, however, not realistic in real world scenarios, see below.

As the colluders do not know fingerprint messages of innocent users, they cannot raise a specific innocent's score to increase his chance to get accused. Instead, they aim to stay undetected by trying to lower their own accusation scores. They succeed, if all colluders get an accusation score lower than the threshold Z .

However, in the real world the colluders are not restricted to this assumption. Instead they can apply attack strategies for which each symbol is chosen also with respect to the symbols observed on all other detectable positions.

According to Definition 21, the colluders may use the information about the symbols y_k , $k = 1, \dots, m$ taken for the other positions $k \neq i$ as well for the selection of the symbol y_i , instead of being restricted to location dependent attack models.

To generalize this observation, the subsequent definitions and statements are needed. Define the accusation score as $S_j = S'_j + S''_j$, with the score from the non-detectable positions denoted as S'_j and S''_j as the score from the detectable positions. The value S'_j is equal for all colluders and cannot be changed due to the marking assumption, see section 4.4. Hence, the optimal attack strategy to avoid getting accused can be defined as in Definition 22.

According to Definition 22: *An attack strategy is referred to as optimal, if it outputs a manipulated fingerprint y^* for a collusion C with arbitrary $j \in C$, that minimizes the probability that the largest colluders' accusation score exceeds the threshold Z .*

$$\Pr[S_{\max} > Z] = \Pr[S''_{\max} > Z - S'_j]$$

As p_i is unknown to the colluders, they cannot compute S''_{\max} exactly. However, using the maximum-likelihood method, the colluders may guess each p_i from their observed symbols. Let b_i denote the number of observed 1's at position i . The maximum-likelihood method results to the estimation $\hat{p}_i = b_i/c$ and

$$\hat{S}''_j = \sum_{i \text{ detectable}} g(X_{j,i}, y_i, \hat{p}_i).$$

The value

$$\hat{y}^* = \arg \min_y (\max_{j \in C} \hat{S}''_j)$$

appears to be a reasonable candidate to minimize $\mathbb{E}[S''_{\max}]$. The subsequent paragraphs describe two heuristics following the stateful attack model that efficiently find good approximations of

\hat{y}^* , and thereby optimal strategies that are to some extent superior to the attack strategies of the stateless attack model.

- **Greedy Attack:** The first heuristic to find $\hat{y}^* = \arg \min_y (\max_{j \in C} \hat{S}_j'')$ is based on a *greedy* approach. Each position is selected consecutively with the symbol that increases the maximal colluder score the least. In the final step, an evolutionary approach is used to further minimize \hat{S}_{\max}'' .

1. **Initialization:** Let $C = \{j_1, \dots, j_c\}$ denote a coalition of size c with corresponding fingerprints X_{j_1}, \dots, X_{j_c} . For simplicity we assume that we have d detectable positions with corresponding indices $\mathcal{D} = \{1, \dots, d\}$. For each $i \in \mathcal{D}$, compute $\hat{p}_i = b_i/c$, with $b_i = \sum_{j \in C} X_{j,i}$ the number of observed 1s.

2. **Loop:** For each $k = 1, \dots, d$, do:

- Set $y_k = 0$ and compute $\zeta_0 = \max_{j \in C} \sum_{i=1}^k g(X_{j,i}, y_i, \hat{p}_i)$
- Set $y_k = 1$ and compute $\zeta_1 = \max_{j \in C} \sum_{i=1}^k g(X_{j,i}, y_i, \hat{p}_i)$
- If $\zeta_0 < \zeta_1$, then set $y_k = 0$. If $\zeta_0 > \zeta_1$, set $y_k = 1$. If $\zeta_0 = \zeta_1$, choose y_k uniformly at random.

3. **Finalization:** The final step loops through all indices in a random order. If a bit-flip reduces the value $\hat{S}_{\max}'' = \max_{j \in C} \sum_{i=1}^d g(X_{j,i}, y_i, \hat{p}_i)$, the corresponding position y_i will be updated. This finalization step terminates when no bit-flip will further reduce \hat{S}_{\max}'' . The forged fingerprint y will be returned.

- **Combinatorial Attack:**

1. *Initialization:* Let $C = \{j_1, \dots, j_c\}$ denote a coalition of size c with corresponding fingerprints X_{j_1}, \dots, X_{j_c} . For simplicity we assume that we have d detectable positions with corresponding indices $\mathcal{D} = \{1, \dots, d\}$. Initialize $\mathcal{U} = \emptyset$.

2. *Find complements:* For each $k = 1, \dots, d$ and $k \notin \mathcal{U}$ do:

- For $l = k+1, \dots, d$ and $l \notin \mathcal{U}$ check if column l is a complement of column k , i.e. $X_{j,k} \neq X_{j,l}$ for all $j \in C$. If so, set $y_k = y_l$ uniformly at random and add k, l to the set U .

3. *Find identical columns:* For each $k = 1, \dots, d$ and $k \notin \mathcal{U}$ do:

- For $l = k+1, \dots, d$ and $l \notin U$ check if column l and k are identical, i.e. $X_{j,k} = X_{j,l}$ for all $j \in C$. If so, set y_k uniformly at random and set $y_l \neq y_k$. Add k, l to the set U .

4. *Finalization:* For the remaining indices $i \in \mathcal{D} \setminus \mathcal{U}$, use the greedy attack strategy from above to finish generating y .

This heuristic uses the symmetry of the score function. That is for example: Assuming the colluders observe the following symbols on two positions i_1 and i_2 :

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

Thus $\hat{p}_{i_1} = 2/5$ and $\hat{p}_{i_2} = 3/5$. By setting $y_{i_1} = y_{i_2}$, it holds $g(X_{j,i_1}, y_{i_1}, \hat{p}_{i_1}) + g(X_{j,i_2}, y_{i_2}, \hat{p}_{i_2}) = 0$ for all colluders $j \in C$. Therefore, when assign these two positions in such a way, the value \hat{S}_{\max}'' remains the same.

The same applies to two identical columns, i.e. $X_{j,k} = X_{j,l}$ for $k \neq l$ and all $j \in C$. To find \hat{y}^* , the following attack applies these two rules to find pairs of complementary and identical columns. The previously described Greedy attack is used for the remaining positions where no such match can be found.

For small collusion sizes, many matching columns can be found. Considering the actual p_i values, the probability that $p_{i_1} \approx p_{i_2}$ is the highest if $\hat{p}_{i_1} = \hat{p}_{i_2}$ and $p_{i_1} \approx 1 - p_{i_2}$ is the highest for $\hat{p}_{i_1} = 1 - \hat{p}_{i_2}$. Therefore, applying the Combinatorial attack strategy, it is expected that such pairs increase the sum S_{\max}'' only marginal.

Note that in the real world, the attackers don't know the actual symbols. However, for watermarking schemes that embed binary watermarks position-wise there will be two groups of colluders for each position, the one with embedded a '1' the other with embedded a '0'. This means, colluders from the one group have other media values than colluders from the other, and throughout the whole media file and fingerprint, these constellation of groups and its inverse constellation are likely to occur.

C Regarding the 2-secure fingerprinting scheme

This section contains the analytical proofs that lead to the zero-false positive error rate of the 2-secure fingerprinting approach presented in chapter 6.1.

In general probabilistic fingerprinting schemes we have to consider two kinds of errors, the false positive (FP) error and the false negative (FN) error, as elaborated in chapter 3. However, as already mentioned, this approach of a 2-secure fingerprinting scheme is not typical for probabilistic code as it provides a provable FP of zero. In other words, it will not occur that the scheme accuses an innocent-fingerprint to have partaken in the collusion.

On the other side, there does exist the probability that the scheme arouses a FN, i.e. it is not able to accuse any colluder at all. Note that since the FP is zero, the FN is equal to not being able to accuse anyone (colluders-fingerprint or innocents-fingerprint) at all. To find out about the amount of FN, more precisely, to find the appropriate upper bound ε_2 on the probability of a FN, we have to consider every possible disadvantageous combination of how the colluder-fingerprints look like, and which attacked fingerprint y they possibly can create.

According to the marking assumption described in chapter 4.4, the colluders are only able to change positions in which their fingerprints differ from each other. Thus, the fingerprint y can be divided into undetectable positions and detectable positions. In figure 18 these parts are denoted as *region A* and *region B* respectively. Here the colluders are denoted as C_1 and C_2 , u_1 denotes an innocent-fingerprint that has a very disadvantageous constellation. This means its Hamming Distance to y is very small.

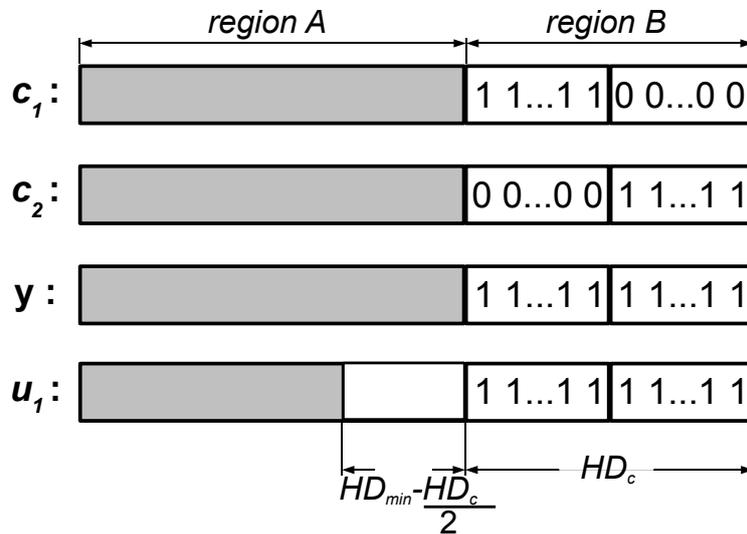


Figure 18: Example for a disadvantageous constellation of 2-secure fingerprints, for which an innocent-fingerprint looks very similar to the manipulated fingerprint y

C.1 The false positive error

In the example of Figure 18 the Hamming Distance of the two colluder-fingerprints $HD(X_{C_1}, X_{C_2})$ is exactly HD_{max} . This obviously is the most disadvantageous constellation for

the content distributor, because the number of detectable positions is the highest possible, and thereby the colluders have the opportunity for the strongest possible collusion attack.

Therefore, let us assume the colluders create the manipulated fingerprint y in a way, that the detected positions show up the same to both. For simplicity we assume they create *region B* consisting of only '1's. Their Hamming Distances to y thus will be exactly $\frac{1}{2}HD_{\max}$. Now there exists a probability that an innocent-fingerprint looks alike u_1 in Figure 18, where u_1 only differs from y in the *white* part of *region A*. This fingerprint's Hamming Distance to y is located as $HD_{\min} - HD_c/2$, where HD_c denotes the actual Hamming Distance between the two colluder-fingerprints. In case there is no appropriate tracing algorithm but only Hamming Distance measurement for the accusation decision, this innocent-fingerprint would get accused misleadingly.

In order to avoid false accusations, we have to consider the set G_1 and G_2 :

$$G_1 := \{X_j; HD(X_j, y) \leq \frac{1}{2}HD_{\max} \forall j \in \{1, \dots, n\}\}$$

$$G_2 := \{X_j; \frac{1}{2}HD_{\max} < HD(X_j, y) \leq \frac{3}{4}HD_{\max} \forall j \in \{1, \dots, n\}\}$$

We want G_1 and G_2 to count as many fingerprints as necessary in such a way that if there is only one colluder-fingerprint appearing in $G_1 \cup G_2$, this fingerprint always has the minimum Hamming Distance to y .

Therefore, we need the following two Lemmas.

Lemma 1 *If the Hamming Distance between two colluder-fingerprints is smaller than or equal to HD_{\max} , then at least one colluder-fingerprint is sorted into G_1 .*

$$HD(X_{C_1}, X_{C_2}) \leq HD_{\max} \Rightarrow G_1 \cap \{X_{C_1}, X_{C_2}\} \neq \emptyset \quad (16)$$

Lemma 2 *If exactly one colluder-fingerprint belongs to $G_1 \cup G_2$, then the Hamming Distance of this colluder-fingerprint to y is smaller than every other fingerprints' Hamming Distance to y .*

$$\exists X_{C_k} \in \{X_{C_1}, X_{C_2}\} : X_{C_k} \notin G_1 \cup G_2 \Rightarrow \exists X_{C_l \neq k} : HD(X_{C_l}, y) < HD(X_j, y) \forall j \in \{1, \dots, n\} \quad (17)$$

Proof of Lemma 1:

Because for all possible pairs of fingerprints $(X_j, X_{j'})$, $j \neq j'$, contained in \mathbf{X} it holds $HD(X_j, X_{j'}) \leq HD_{\max}$, see Definition 23. Thus it also holds for the colluder-fingerprints (the left side of equation (16) is always true). For one index $k \in \{1, 2\}$ it must hold $HD(X_{C_k}, y) \leq 0.5 HD_c$. Consequently, the Hamming Distance of this colluder-fingerprint must be smaller than (or equal to) $0.5 HD_{\max}$, thus it is automatically sorted into G_1 . \square

Proof of Lemma 2:

Let λ_1 and λ_2 be the parts of y (more precisely of region B) solely contributed by X_{C_1} and X_{C_2} , respectively, i.e. the part where X_{C_1} and X_{C_2} differ from each other. Thus it holds $\lambda_k \in [0, 1]$ and $\lambda_1 + \lambda_2 = 1$.

The left side of equation (17) means that there is one colluder-fingerprint not sorted into $G_1 \cup G_2$. Thereby without loss of generality and with the definition of G_2 we assume it is X_{C_2} . Thus it holds $HD(X_{C_2}, y) > \frac{3}{4}HD_{\max}$. Consequently $\lambda_2 HD_c > \frac{3}{4}HD_{\max} \geq \frac{3}{4}HD_c$ which means $\lambda_2 > \frac{3}{4}$ and thus $\lambda_1 < \frac{1}{4}$. Hence it holds

$$HD(X_{C_1}, y) = \lambda_1 HD_c < \frac{1}{4}HD_c \quad (18)$$

Thereby we induce a *reductio ad absurdum* (proof by contradiction). Let us assume without loss of generality:

$$\exists X_j, j \in \{1, \dots, n\}, j \neq C_k : HD(X_j, y) < HD(X_{C_1}, y) \quad (19)$$

We consider three different cases:

1. y and X_j differ only in region A.
2. y and X_j differ only in region B.
3. y and X_j differ in both region A and region B.

For simplicity we write the parts of the fingerprint in which X_j differs from y as $\lambda_j HD_c$ with $\lambda_j < \lambda_1$. Thereby $HD(X_j, y) < HD(X_{C_1}, y)$ is ensured.

Proof of 1.: From the condition in 1. it follows

$$HD(X_j, X_{C_1}) = \lambda_j HD_c + \lambda_1 HD_c < 2\lambda_1 HD_c < 2 \cdot \frac{1}{4}HD_c = \frac{1}{2}HD_c \leq \frac{1}{2}HD_{\max} \quad (20)$$

The Hamming Distance condition from definition 23 in chapter 6.1 prescribes that the Hamming Distance between two feasible fingerprints of \mathbf{X} is at least HD_{\min} . But the same condition also prescribes $\frac{1}{2}HD_{\max} < HD_{\min}$. Thus we have a contradiction meaning that for this case there cannot exist an innocent-fingerprint X_j with a Hamming Distance to y smaller than X_{C_1} .

Proof of 2.: From the condition in 2. it follows

$$HD(X_j, X_{C_1}) = \lambda_j HD_c + \lambda_1 HD_c < 2\lambda_1 HD_c < 2 \cdot \frac{1}{4}HD_c = \frac{1}{2}HD_c \leq \frac{1}{2}HD_{\max}$$

This is equal to equation (20), the only difference in this case is that both parts representing the Hamming Distance to y , i.e. $\lambda_j HD_c$ and $\lambda_1 HD_c$, may overlap. However, this means their distance to each other decreases and therefore (20) holds as well. Consequently, we get the same contradiction as above saying that there cannot be an innocent-fingerprint X_j with a Hamming Distance to y smaller than X_{C_1} .

Proof of 3.: Let a be the part of region A in which X_j and y differ and let b be the corresponding part of region B. Thus it holds $a + b = \lambda_j HD_c$. In analogy to (20):

$$HD(X_j, X_{C_1}) = a + \lambda_1 HD_c + b = (\lambda_1 + \lambda_j)HD_c < 2 \cdot \frac{1}{4}HD_c \leq \frac{1}{2}HD_{\max}$$

Again, the case for which b and $\lambda_1 HD_c$ overlap would even decrease $HD(X_j, X_{C_1})$. Thus, there is a contradiction also for this case.

In summary, due to the contradictions above, the assumption $HD(X_j, y) < HD(X_{C_1}, y)$ must be wrong and Lemma 2 holds true. \square

This allows us to prove theorem 1 from chapter 6.1 or equivalent:

- Theorem 3**
1. *If there is associated exactly one pair of fingerprints as possible colluder pair, and at the same time there is no fingerprint with a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$, this possible colluder pair is the colluder-fingerprints.*
 2. *If there is associated more than one pair of fingerprints as possible colluder pair and one fingerprint appears in all possible colluder pairs and at the same time there is no different fingerprint with a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$, this fingerprint is a colluder-fingerprint.*
 3. *If no pair of fingerprints is assigned as possible colluder pair, the fingerprint with the minimum Hamming Distance to y is a colluder fingerprint.*

Proof of Theorem 3.1 From equation (18) from the proof of lemma 2 it follows, that if one colluder-fingerprint was not sorted into $G_1 \cup G_2$, the other colluder-fingerprint must have a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$. Thus the corresponding condition in theorem 3.1 automatically ensures that both colluder-fingerprints must appear in $G_1 \cup G_2$. Together with lemma 1 (one colluder-fingerprint must appear in G_1) this means the one found possible colluder pair is the colluder-fingerprints. \square

Proof of Theorem 3.2 In analogy to the proof of 3.1, if no fingerprint has a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$, the colluder-fingerprints are found as a possible colluder pair. In case there are several pairs of fingerprints identified as possible colluder pairs, the one fingerprint that appears in all those pairs obviously appears as well in the pair of colluder-fingerprints and thus must be one of the colluder-fingerprints. \square

Proof of Theorem 3.3 This follows directly from Lemma 2. If there is not found a possible colluder pair, one of the two colluders does not appear in $G_1 \cup G_2$, hence the fingerprint with the minimum Hamming Distance to y is a colluder-fingerprint. \square

From theorem 3 we can directly reason the following corollaries:

Corollary 2 *If there is associated exactly one pair of fingerprints as possible colluder pair, and at the same time one fingerprint of this pair has the minimum Hamming Distance to y , this fingerprint is a colluder-fingerprint.*

Corollary 3 *If there is only one colluder-fingerprint within $G_1 \cup G_2$, he is closest to y , and thus, only this colluder can get accused.*

Otherwise, the searching algorithm sorts out the combinations of innocent-fingerprints that come into consideration.

From this follows that the set of fingerprints to be considered for parent pair search does not need to contain more suspects than claimed for $G_1 \cup G_2$. In the case $HD(X_{C_k}, y) \approx HD_c/2$ for $k \in \{1, 2\}$ both colluders are within. If one colluder-fingerprint shows up to y less than the other, he might fall out of $G_1 \cup G_2$, but the other's chance to be caught increases decisively.

Table 22: False negative error in different settings

fingerprints n	code length m	"strategy"	FP	FN (straight)	FN (pair search)
10^2	50	0.1	0	0.0004	0.0004
		0.3	0	0.0014	0.0014
		0.4	0	0.0081	0.0051
		0.5	0	0.0482	0.0117
10^3	100	0.01	0	0	0
		0.1	0	0	0
		0.3	0	0	0
		0.4	0	0	0
		0.5	0	0.0015	0
10^3	130	0.01	0	0	0
		0.1	0	0	0
		0.3	0	0	0
		0.4	0	0	0
		0.5	0	0	0

The parent pair searching algorithm then sifts the responsible fingerprints. Due to the prescribed Hamming Distances this searching algorithm finds out the one combination which has been necessary to create y . If there are more possible combinations found and the algorithm cannot assign a definite responsible, no one gets accused. Therefore see about the false negative error rate discussed below. It is not possible to accuse an innocent pair. Hence the false positive error rate is zero.

C.2 The false negative error

Since we propose a scheme strongly avoiding false positives, i.e. wrong accusations, we have to consider the probability that the scheme is not able to accuse anyone at all, i.e. the probability for the false negative error (FN).

Note that for the cases in which the part from one colluder-fingerprint erratically exceeds the one from the other, so that the second is not considered anymore in G_1 or G_2 , the false negative error rate is zero. This is because these cases – that will be associated to theorem 3.3 – fall out of any calculations, as one (colluder-)fingerprint is always suspected. Results of attacks in which the colluders do not partake equally are depicted in table 22.

Hence, the focus is on cases in which the colluders partake approximately equally in the creation of y . These correspond to the results with a strategy of 0.5, mirroring that each colluder-fingerprint contributes 1/2. A false negative error occurs, if we cannot accuse one possible colluder pair because there are at least two disjoint possible colluder pairs found.

In practice, a relaxation of the tracing algorithm is capable to further improve the results for the false negative error rate. In the cases where the group G_1 contains only two fingerprints and these two are as well the only possible colluder pair found, directly outputting these two further reduces the events of false negative error significantly. At the same time we still meet the zero false positive error rate. However, an analytical proof that the false positive error remains zero

Table 23: False negative error in Nuida’s setting [79], both colluders participating equally (0.5)

fingerprints n	code length m	"strategy"	FP	FN (straight)	FN (pair search)
10^2	61	0.5	0	0.0018	0 (1)
	71	0.5	0	0.0044	0.0002
	91	0.5	0	0.0004	0
10^3	74	0.5	0	0.033	0.0021
	83	0.5	0	0.011	0.0005
	93	0.5	0	0.0044	0.0001
	102	0.5	0	0.0021	0.0001

at any time is not found yet. The results for the false negative error can be seen in table 22. Those for the relaxation of the tracing algorithm are referred to as FN (*parent pair*) compared to those for the straight version as described in chapter 6.1 denoted as FN (*straight*).

Besides the further reduced false negative error rates for the relaxed version of the tracing algorithm, we also depict results with extremely short fingerprinting codes with lengths of 50 bits for a code of 100 fingerprints. As far as we know there are no comparable short code length providing collusion security.

For reasons of comparison, table 23 again shows the results for *Nuida’s* setting [79] already depicted in chapter 6.1, but now also with the relaxed version as described above.

C.3 Error probabilities for collusions larger than two

In practice, the question might arise about what happens in case there are three (or more) customers collaborating to create the forgery with the manipulated fingerprint y . For the Tardos codes [114] and its derivatives including the other approaches presented in this thesis a collusion larger than the maximum expected collusion size c_0 has only minor effect on the false positive error rate. The major effect it has on the false negative error rate. This is acceptable, because not being able to find any colluder is to be preferred over accusing an innocent. The 2-secure fingerprinting approach presented in chapter 6.1 provides similar results, although, it is not at all belonging to the class of Tardos codes. For the same test sets as mentioned above, but with actual collusion sizes of $c = 3, 4, 5$ we still never accused any innocent fingerprint ($FP=0$). However, the false negative error rate was significantly increased. The highest false negative error rate in each case was received with a collusion "strategy" of 0.5 (colluders partake equally in the forgery) with a rate close to 1.0, i.e. in almost all attacks no fingerprint was output by the tracing algorithm. We find two reasons for this behavior: First, for the Tardos codes, the optimal functions g_1 and g_2 to create the accusation score, see chapter 4.5, for a collusion of two ($c_0 = 2$) are indeed represented by the Hamming Distance as proven in [103]. For this reason our 2-secure fingerprinting approach, for which the Hamming Distance is taken as some kind of scoring function as well, behaves similar to the Tardos codes. Second, to actually reason the zero false positive error rate, we need to analyze step by step the decisions made by tracing algorithm. In more than 95 percent of the setting from above, there were at most two fingerprints sorted into G_1 . However, none those ever had a Hamming Distance to y smaller than $\frac{1}{4}HD_{\max}$. In the further proceeding of the tracing algorithm, no possible colluder pair was found. Given these observations, the tracing algorithm – assuming a collusion of two – concludes:

- at least one of the fingerprints in G_1 must be a colluder-fingerprint (lemma 1)

-
- as no possible colluder pair was found, only one of the fingerprints in G_1 is a colluder-fingerprint and its counterpart is not within the union set of G_1 and G_2 .
 - the one with the smallest Hamming Distance to y must be a colluder-fingerprint (3)

However, as no fingerprint provides a Hamming Distance to y that is smaller than $\frac{1}{4}HD_{\max}$ including the one with the smallest Hamming Distance to y , this is a contradiction to findings from the proof of lemma 2. The consequence is that we rather not accuse any fingerprint at all.

In short, the false negative error rate increases decisively when the collusion exceeds two participants, but – more important – the approach still provides a zero false positive error rate.

For continuing examples of the application of this approach see for instance [101].

D Regarding the 3-secure fingerprinting scheme

The 3-secure tracing algorithm proposed in chapter 6.2 is very simple structured for reasons of adaptivity. In this section we present some more possibilities for this tracing algorithm, more precisely for the minority tracing strategy. Before that some more empirical test results are given.

As the value which decides which tracing strategy to take is only a rough proposition, sometimes the 'wrong' tracing strategy is selected. This leads to imprecisenesses for the code, yet the impact is very low. In table 24 one recognizes that the minority vote attack strategy nearly always achieves the lowest false alarm (FP) rate. This is what we expected, because the minority vote tracing strategy is special tailored for the minority vote attack and it is by construction more sophisticated (and requires more computational effort) than the majority tracing strategy.

Compared to the other attack strategies, the random vote strategy achieves a higher false negative error rate (the percentage where no accusation is possible). This is because of the fact, that both tracing strategies are especially focused on the corresponding attack strategy and strongly try to avoid false positives whatever the attack. Hence, this goes ahead with an increased false negative error rate. However, it remains far lower than the error bound selected in the fingerprint generation process.

Note that the results posted in this table reflect a detect one scenario, i.e. only accusing the one colluder suspected the most, in order to mirror the specified algorithm as presented in chapter 6.2.

Table 24: Comparison of the behavior of the code at different attack strategies

	code length	number of fingerprints	Attack Strategies		
			minority vote	random	majority vote
detection rate	250	100	99.998%	>99.93%	100%
false negatives (FN)	250	100	<0.002%	<0.07%	0%
false positives (FP)	250	100	< 10^{-6} %	$2 \cdot 10^{-6}$ %	< 10^{-6} %
detection rate	250	1000	>91%	88.25%	>99,99%
false negatives (FN)	250	1000	<9%	11.75%	<0.01%
false positives (FP)	250	1000	< 10^{-5}	< 10^{-5} %	< 10^{-5} %

The minority tracing algorithm stops after a colluder triple was found within one pair of PG^l and one pair of PG^r or if each pair of PG^l has been considered with each pair of PG^r .

Instead of stopping the algorithm after a colluder triple was found, one could continue for instance as follows to get a lower false alarm rate: First take the fingerprint of the fingerprint appearing in both pairs PG^l and PG^r of the colluder triple and add a pair of PG^l or PG^r (from first level upwards) with at least one different fingerprint than in the colluder triple. Check for those if y also belongs to their envelope ϵ and thus they themselves form a colluder triple. Continue this way for both PG^l and PG^r until a certain threshold (to be chosen by the distributor) is reached, or a *sufficient* number of colluder triples are found, or all pairs have been considered.

In case the results are still not satisfying, one could continue as follows: Take the those two fingerprints of the colluder triple that appeared only once in the according pairs PG^l and PG^r

and search within both groups G^l and G^r for a different third fingerprint and check if the corresponding triple could have created y , and thus forms a further colluder triple. This is continued for each fingerprints in G^l and G^r .

Finally we should have found one suspected colluder triple from the algorithm presented in chapter 6.2 and possibly some other colluder triples during the two expansion steps just explained. Next we simply search for the fingerprints occurring in all colluder triples found. These are assumed as having pertained in the collusion and are output by the tracing algorithm.

Another possibility is to follow the idea of the majority tracing strategy and output the fingerprint appearing in *most* colluder triples, instead of the ones appearing in *all* colluder triples.

E Regarding the fingerprint matrix generation

In chapters 7.2, 7.3 and 8.2 we proposed optimizations and adaption for the generation of the fingerprint matrix \mathbf{X} . In the following we expand the considerations and reasons for its necessity.

Analyzing the fingerprint generation process due to [65] it turns out, that some fingerprint matrices result in much higher error rates than others, as explained in [88]. In chapter 7.2 we identified two major statistical imprecisenesses that cause these variations: the unfavorable distribution of significant positions in the column generation and the asymmetric distribution of '1's and '0' in the rows of the fingerprint matrix \mathbf{X} . Both properties and the reason for their appearance are analyzed in the following. We selected the detect one scenario (only the most suspected fingerprint is output) for reasons of simplicity and because the detection scenarios do not influence the statistical properties of the generation process nor the accusation scores to be analyzed.

In [88] we observed that the error rate in the detect-one scenario varies heavily with different underlying fingerprint matrices. Some are especially unfavorable leading to 50 times higher error rates than other matrices. Figure 19 depicts the attack strategy that leads to the highest (minority vote) and lowest (minimal mutual information attack) variances regarding the error rates.

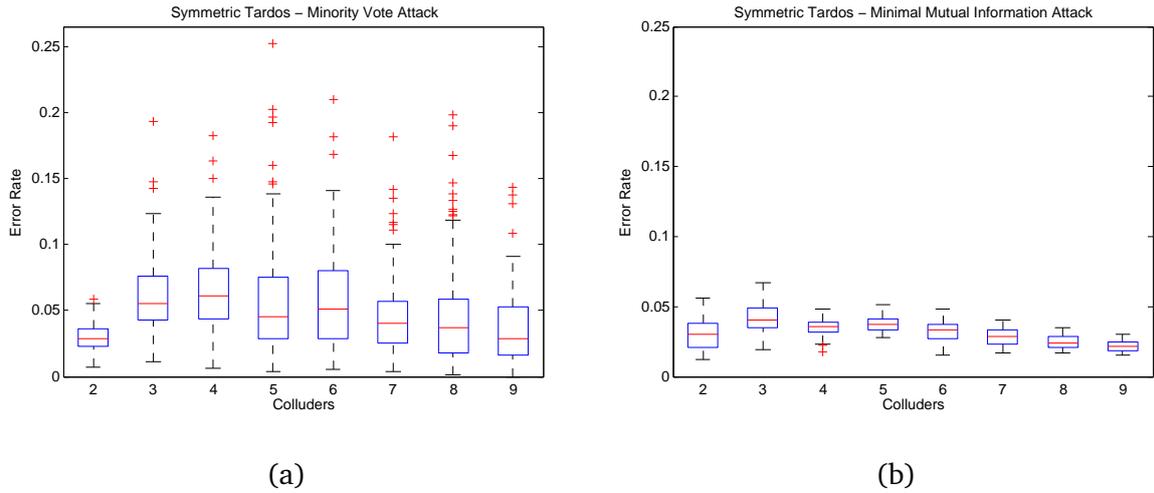


Figure 19: Box plot of the error rates for (a) minority vote attack and (b) minimal mutual information attack for the Tardos scheme with continuous arcsine-distribution. Based on [88]

For the tests we chose the code length as $m = \lceil d_m c_0^2 \ln(1/\varepsilon_1) \rceil$ with a cutoff $t = 1/(d_t c_0)$ according to the recently published approach in [64] that proves to be optimal, see appendix A. The values for the code length m and the cutoff t can be found in table 25. Note that setting a threshold Z is not necessary for the detect-one scenario, as solely the fingerprint with the highest accusation score is output.

We simulated different attacks for $n = 1000$ fingerprints and for $c = c_0 = 2, \dots, 9$ colluders. For each new value of c_0 and each attack strategy 100 different fingerprint matrices were generated. For each matrix, 1000 attacks with a random set of colluders were executed. In summary this results in 100.000 attacks per attack strategy and per collusion size for the detect one scenario.

Table 25: Parameter selection according to [64] for $n = 1000$, $\varepsilon_1 = 0.1$ and $\varepsilon_2 = 1/2$ and the code length as $m = \lceil d_m c_0^2 \ln(1/\varepsilon_1) \rceil$.

c_0	2	3	4	5	6	7	8	9
d_m	11.931	10.818	10.149	9.689	9.346	9.078	8.860	8.678
m	110	225	374	558	775	1025	1306	1619
d_t	27.223	28.930	30.325	31.519	32.586	33.546	34.424	27.223

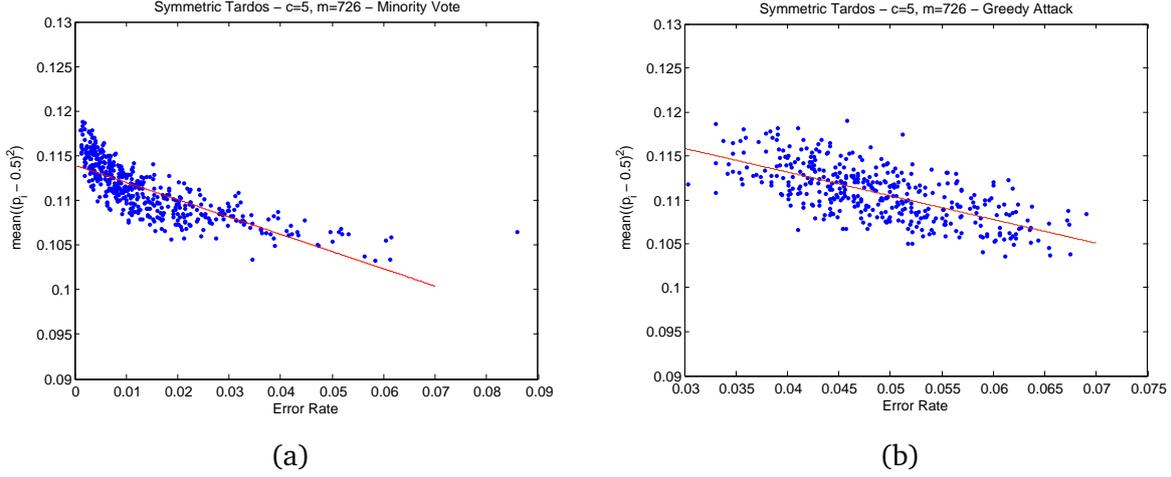


Figure 20: Squared distance of the probability values p_i from the center 0.5 in comparison to the error rate. Detect one scenario with $n = 1000$ fingerprints, $c = 5$ colluders, and $m = 726$ code length. (a) Minority vote attack, (b) Greedy attack. The red line is the trend. Continuous arcsine-distribution with cutoff $t = 1/(31.52c_0)$ was selected. Based on [88]

The highly varying variances of the errors are caused by statistical properties of weak fingerprint matrices. These are analyzed in the following. We generated 400 different fingerprint matrices using the arcsine-distribution as explained in 4.5. The parameters were selected as $n = 1000$ fingerprints, $c_0 = c = 5$ colluders and a code length of $m = 726$. Each matrix was attacked with the minority vote attack, as this attack leads to the highest variances. The attacks were conducted by 20.000 collusions. We also conducted Greedy attack from the stateful attacks, that was investigated in chapter 4.6.2.

Figure 20 mirrors the statement of chapter 7.3, saying that matrices with higher error rates are usually more centric, i.e. usually have more probability values p_i close to 0.5. A consequence is that with probability values close to 0.5, the probability that the colluders may detect this position (see definition 16 for detectable positions in chapter 4.4) is higher.

The second identified property is the difference between the expected number of ones in each column and its actual number. Given p_i , the expected number of ones in column i would be np_i . Let λ_i denote the actual number of ones in column i . We computed the *Pearson* correlation (see for instance [89]) between the error rate and statistical properties of the fingerprint matrix \mathbf{X} in a minority vote attack and a Greedy attack, see chapter 4.6. The results are depicted in table 26. Combining this results with the analysis above we get that if the difference between

Table 26: *Pearson* correlation between the error rate and statistical properties of the fingerprint matrix X for the Minority Vote Attack and the Greedy Attack. The first statistical property describes the centrality of the bias values. The second describes the difference between expected and empirical mean, where λ_i denotes the column sum for column i .

Correlation calculation	Minority Vote	Greedy Attack
$\frac{1}{m} \sum_{i=1}^m (p_i - 0.5)^2$	-0.778	-0.668
$\frac{1}{m} \sum_{i=1}^m (p_i - \lambda_i/n)^2$	0.285	0.247

np_i and λ_i increases, i.e. a column has a lot more or less '1's than expected, the observed error rate also increases.

To avoid such weak matrices, a content distributor could generate a fingerprint matrix and determine the empirical error rate. In case it exceeds a specific limit, the matrix will be neglected and another matrix will be generated. However, this is not very practical. Therefore, chapters 7.2, 7.3 and 8.2 propose optimizations of the fingerprint generation process in order to a priori avoid generation of weak matrices.

F Regarding the discrete distribution

Chapter 7.3 proposed a new fingerprint generation process using discrete distributions. In the following we will expand the explanations and its rationale and present some more evaluations.

F.1 Continuing explanations

Section 7.2 already discussed that some fingerprint matrices lead to particularly high error rates. For the discrete distribution according to [65] this error rate is up to 50 times higher for disadvantageously generated matrices than for other *average* matrices. Two statistical properties were identified that these matrices had in common: The centrality of the probability values, i.e. a lot of probability values p_i are close to the center 0.5, and the difference between the probability value p_i and the empirical mean for this column λ_i/n , with λ_i denoting the number of ones in the column i . Both properties give an advantage to colluders, for example the amount of detectable positions increase when a lot of probability values are close to 0.5, see appendix E.

The adaption presented in chapter 7.3 avoids the generation of weak fingerprint matrices and thus reduces the average error rate, as shown in [88].

Explanations to the three steps of the fingerprint generation algorithm are given below.

1. *Avoid centrality of the probability values*

The discrete distribution according to [65] proposes $T = \lceil c_0/2 \rceil$ different values, each with the same probability $1/T$. As a first step we ensured that each of these values occur with the same frequency in the probability vector $\mathbf{p} = (p_1, \dots, p_m)$, see chapter 4.3. For $T = 2$, half of the probability values would be p_1 and the other half p_2 accordingly.

In case m is not divisible by T , it is ensured that each probability value occurs at least $\lfloor m/T \rfloor$ times and at most $\lceil m/T \rceil$ times within \mathbf{p} . A randomly shuffling of \mathbf{p} hinders that a collusion guesses a correct probability value p_i .

2. *Ensure column sums match expected column sums*

In the classical fingerprint generation method according to Tardos [114], each position in the fingerprint matrix is generated independently with $\Pr[X_{i,j} = 1] = p_i$, see chapter 4.

As discussed in appendix E, this may lead to a varying number of '1's in a column, differentiating much from the expected number. For example, if $p_i = 0.6$, the number of '1's in column i should be around 60%. If the number of '1's is around 55% only, the matrix is likely to result to higher error rates.

Step 2. of the algorithm in chapter 7.3 tangles this instability by enforcing a '1' in $\lfloor np_i \rfloor$ positions. With probability $np_i - \lfloor np_i \rfloor$ an additional entry in the respective column is set as '1'. This ensures that the number of '1's in column i is np_i . For obvious reasons these '1's are spread randomly over the different fingerprints and the remaining positions are filled with '0'.

3. *Ensure row sums match expected row sums*

The third optimization step ensures that the number of '1's in each row does not differentiate too much from the expected number, which was identified as disadvantageously, see appendix E.

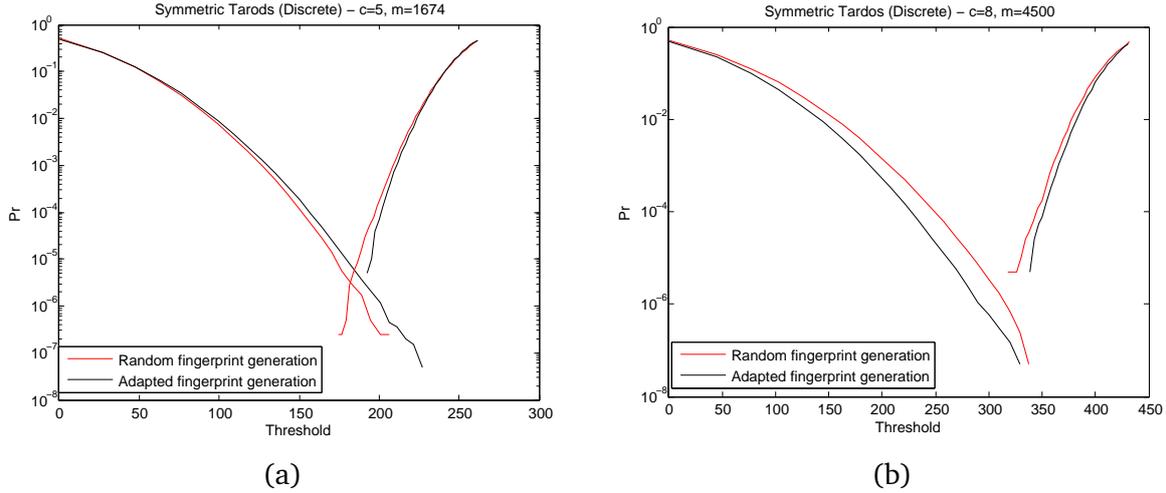


Figure 21: False positive error rates (left lines) for minority vote attacks and false negative error rates (right lines) for the Combinatorial attack (appendix B) with dependence on the threshold for (a) $c_0 = c = 5$ colluders and code length $m = 1.500$ and (b) $c_0 = c = 8$ colluders and code length $m = 4.500$. Based on [88]

For arbitrary sizes of T , this step can be formalized as follows. Be $\mathcal{S}_p = \{i | 1 \leq i \leq m, p_i = p\}$ the index sets of the columns with probability value p . The sets \mathcal{S}_p are comparably large. This is because the discrete distribution according to [65] has only $T = \lceil c_0/2 \rceil$ different probability values and $m \gg T$. Let X_{j, \mathcal{S}_p} denote the positions \mathcal{S}_p of fingerprint X_j . The number of expected '1's on these positions is $p \cdot |\mathcal{S}_p|$. Ensure, that each fingerprint contains approximately $\lceil p \cdot |\mathcal{S}_p| \rceil$ '1's and approximately $\lceil (1-p) \cdot |\mathcal{S}_p| \rceil$ '0's in the different sub-fingerprints X_{j, \mathcal{S}_p} for all p .

F.2 Continuing evaluations

Lots of empirical test were conducted to receive the results presented in chapter 7.3 and below. The following only describes an excerpt of the results and evaluations examined in [88]. The adapted fingerprint generation method is evaluated applying the symmetric score function of [121] presented in chapter 4.5.

Here, we present tests for collusion sizes $c = c_0 = 2, \dots, 9$. For all attack strategies except the minority vote attack, the false positive error rate for the adapted fingerprint generation method of chapter 7.3 stayed much below the required bound ε_1 . However, the minority vote attack, which is the false positive maximizing attack strategy, see chapter 4.6, resulted in slightly higher false positive error rates for $c \leq 6$, while for $c > 6$ the approach of chapter 7.3 resulted in significant smaller false positive error rates. For the false negative error rate, we observed no significant differences for all attack strategies and all evaluated collusion sizes. In figure 21 is illustrated the false positive and the false negative rate for $c_0 = c = 5$ and $c_0 = c = 8$ colluders. The lines on the left hand side in each figure depict the probability that an innocent fingerprint's score exceeds the given threshold, i.e. the false positive rate for a single innocent fingerprint. The lines on the right-hand side represent the probability that all colluder-fingerprints got a score below the threshold, i.e. the false negative error rate.

The results depicted in table 10 in chapter 7.3 were received as follows. To determine the required code length m with which we achieve certain false positive rates $\hat{\varepsilon}_1$ and false negative

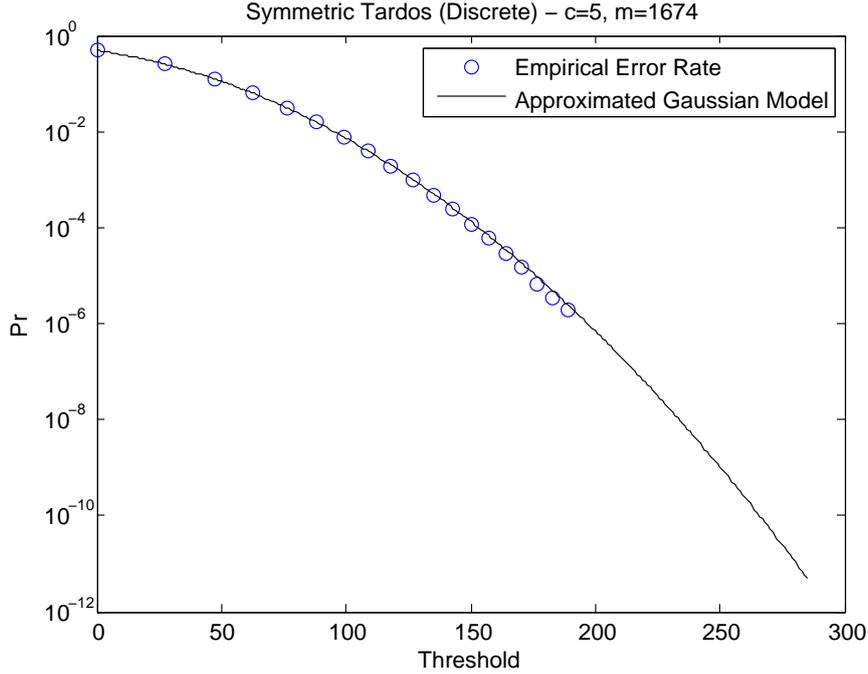


Figure 22: Empirically measured false positive rate and the approximated Gaussian model with $a = 0.726$, $b = -39.96$ and $c = 65.52$ with dependence on the threshold for $c = 5$ colluders and a code length of $m = 1674$. Based on [88]

rates $\hat{\epsilon}_2$, we first measured the empirical error rates for various collusion sizes and various code lengths. The false negative rate was determined by averaging over two million attacks and the false positive rate was measured generating 20 million innocent fingerprints, each time with different probability values, and executing a collusion attack for each fingerprint.

Next, we determined the threshold with which we achieved a false negative rate of $\hat{\epsilon}_2 = 10^{-3}$ for the Greedy attack. For this threshold, the false positive rate $\hat{\epsilon}_1$ is determined in the following way. As the false positive rate might be extremely small, e.g. $\hat{\epsilon}_1 = 10^{-10}$, it cannot directly be derived from the results of 20 million executed attacks (because with a demanded error probability of $\hat{\epsilon}_1 = 10^{-10}$, there might not occur a false positive error at all within 20 million attacks). Therefore, a Gaussian model with $\hat{f}(Z) = a \exp(-((Z-b)/c)^2)$ and $a, b, c \in \mathbb{R}$ was determined which approximates the false positive rate for any given threshold Z (for the Gaussian model see for instance the corresponding explanations in [121]). The model was determined for thresholds with which we received false positive rates between $\hat{\epsilon}_1 = 10^{-1}$ and $\hat{\epsilon}_1 = 10^{-6}$. An example of such a derived Gaussian model is depicted in Figure 22.

The example above with $c = c_0 = 5$ and $m = 1674$ and a threshold of approximately $Z = 211$ results in a false negative error of $\hat{\epsilon}_2 = 10^{-3}$ for the adapted fingerprint generation method of chapter 7.3. With this threshold, we constructed the function $\hat{f}(Z)$ as described above and thereby we approximated the false positive rate as $\hat{\epsilon}_1 = \hat{f}(211) \approx 3.09 \cdot 10^{-7}$.

This way we tested for several collusion sizes and code lengths. Figure 12 depicts the resulting false positive error rate for 4, 6, and 8 colluders, where the false negative rate is fixed as $\hat{\epsilon}_2 = 10^{-3}$. The figure tells, that the false positive error rate decreases nearly linear in $\ln(1/\hat{\epsilon}_1)$ for the different code lengths tested. Thereby we can parameterize the code length as $m = d_m c_0^2 (\ln(1/\hat{\epsilon}_1) + \kappa)$. The parametrization for the tested collusion sizes were listed in table 10.

This means, a scheme with the adapted fingerprint generation method and code length of $m = 3.93c_0^2(\ln(1/\hat{\epsilon}_1 + 0.09))$ will result in a false positive error rate of about $\hat{\epsilon}_1$, if the threshold is set in a way that the false negative error rate is $\hat{\epsilon}_2 = 10^{-3}$.

As figure 12 and table 10 report, the adapted fingerprint generation method investigated in chapter 7.3 result in no improvement at all for $c_0 \leq 6$. However, for $c_0 > 6$ the approach leads to much lower error rates. Equivalent, lower error rates allow the content distributor to choose shorter code lengths and still achieve the selected security level.

G Regarding the dynamic threshold

This section expands the evaluations of the tracing algorithm with dynamic threshold calculation presented in chapter 7.4.

The interleaving score function proposed in [86] shows the potential to drastically reduce the code length for a *Tardos*-based fingerprinting scheme. It was shown, that the interleaving score function optimizes the performance indicator $\tilde{\mu}_c/\tilde{\sigma}_{\text{inn}}$ against arbitrary attack strategies for a continuous arcsine-distribution with no cutoff. However, as examined in [88], if the cutoff t is too small or even $t = 0$, the scheme results in comparably high false positive error rates. This is the case if the skewness of the distribution of the innocent-fingerprints' scores is positive. Also, if it holds $t \rightarrow 0$, the skewness converges towards infinity. Therefore, in practical scenarios the cutoff must be selected sufficiently large.

A further downside of the interleaving score function is that the standard deviation σ_{inn} of the innocent-fingerprints' scores heavily depends on the collusion strategy. That is why accusing all fingerprints with a score exceeding a predefined fixed threshold Z is no longer feasible.

Alternatively, we can compute $\hat{\sigma}_{\text{inn}}$ and accuse all those fingerprints, whose score exceeds an adjusted threshold $S_j > Z\hat{\sigma}_{\text{inn}}$, for some predefined threshold Z . Equivalent we can normalize the scores and accuse all fingerprints with $S_j/\hat{\sigma}_{\text{inn}} > Z$.

The effect of the positive skewness can be seen once more in figure 13 in chapter 7.4. It depicts, that the false positive rate decreases slower than $1 - \Phi(Z)$, with $\Phi(Z)$ the normal cumulative distribution function representing the actual expected error rate, see [121] and [88].

Figure 13 (b) depicts the probability that all colluder-fingerprints get scores below the threshold, i.e. the probability of a false negative error. The false negative error rate heavily depends on the applied attack strategies. For $c_0 \leq 4$ the minority vote attack leads to the highest false negative rate. For $4 < c_0 < 10$ the minimal mutual information attack of *Furon* and *Pérez-Freire* [41], see chapter 4.6, leads to the highest false negative rate. Though optimal in an asymptotic sense, for the tested values of c_0 the interleaving attack performed comparably bad.

In figure 23 we list the worst case attacks for the symmetric score function and for the interleaving score function. The collusion size was selected as $c_0 = c = 5$ and $c_0 = c = 9$. For reasons of comparison, the scores have been normalized, i.e. $S_j/\hat{\sigma}_{\text{inn}}$ for the interleaving score function and S_j/\sqrt{m} for the symmetric score function.

One recognizes that for the symmetric score function the false positive rate decreases faster. This matches our expectations, and is again reasoned by its negative skewness, compared to the positive skewness of the interleaving score function. However, with the interleaving score function the scores of the colluder-fingerprints are significant larger, enforcing the choice of a larger threshold Z . For example, a threshold of $Z = 4.4$ results in false negative rates of about 10^{-3} and false positive rates of about $4.8 \cdot 10^6$ for the symmetric score function. Whereas a threshold of $Z = 5.8$ achieves false negative error rates of about 10^{-3} and more than 60 times smaller false positive error rates of about $7.9 \cdot 10^{-8}$ for the interleaving score function. For $c_0 \geq 9$ the difference further increases, i.e. the interleaving score function results to much lower error rates when applying the dynamic threshold computation presented in chapter 7.4.

Figure 24 gives an impression of the superiority of the interleaving score function compared to the symmetric score function with respect to our dynamic threshold calculation of chapter 7.4. With fixed false negative error rate of $\hat{\epsilon}_2 = 10^{-3}$, it shows results for the false positive error

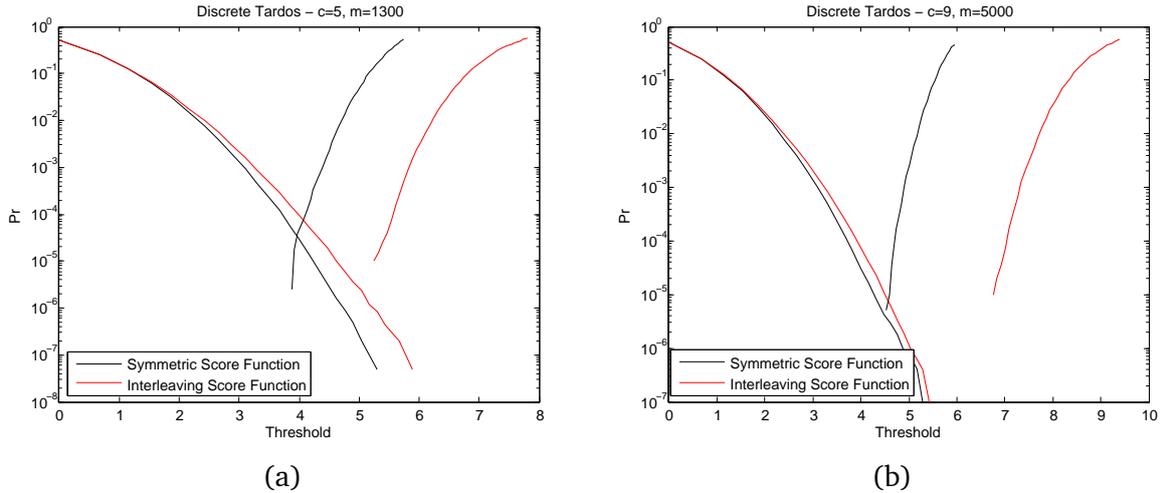


Figure 23: Maximal observed false positive error rate and false negative error rate for the symmetric score function and the interleaving score function with dependence of the threshold. (a) $c_0 = c = 5$ and $m = 1.300$, (b) $c_0 = c = 9$ and $m = 5.000$. The scores are normalized. Taken from [88]

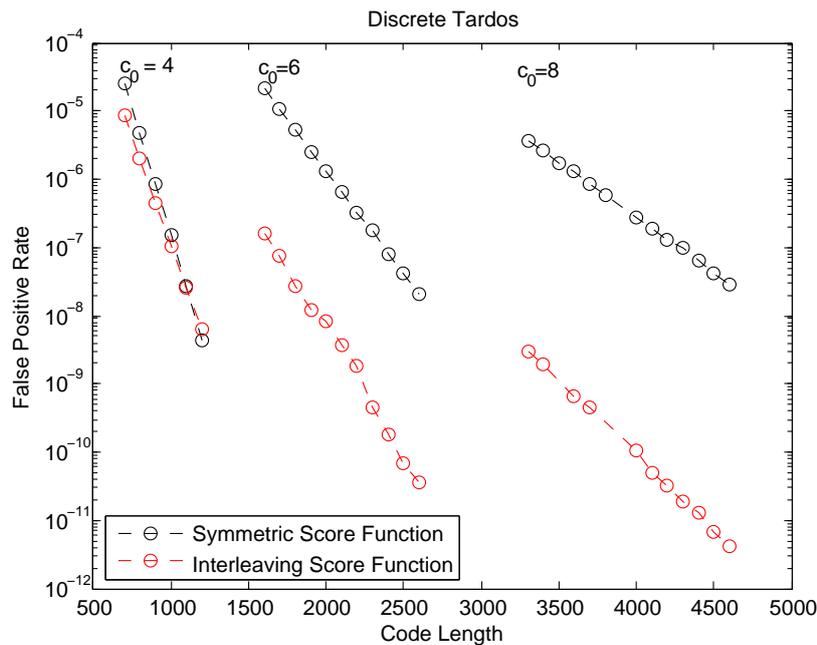


Figure 24: False positive error rates for the symmetric score function and for the interleaving score function with the proposed dynamic threshold calculation method for 4, 6, and 8 colluders. Taken from [88]

rate at various code lengths. Especially for larger collusion sizes, the interleaving score function outperforms the symmetric score function significantly.

This gives reason for further considerations on the threshold calculation for the interleaving score function. As a consequence, *Ibrahimi et al.* [55] recently published an adjusted threshold calculation for the interleaving score function (However it was intended for very large collu-

sions). Shortly afterwards, *Laarhoven* [62] published an approach proposing a modified version of the interleaving score function, also proving to be optimal in an asymptotic sense and this time no cutoff t (see for instance chapter 4.5) is needed any more. Though at the very end of this thesis work, we took on those approaches and elaborated their performances in concordance with the dynamic threshold calculation presented in chapter 8.3.

H Regarding the joint decoding approach

Our joint decoding fingerprinting approach presented in chapter 8.1 outperforms comparable approaches in terms of complexity by construction. We showed that it also performs well regarding the error rates and code length. More tests to other approaches confirm this statement. In the following we present some more test and considerations regarding the joint decoding approach. Before that we give an explanation of the stateful attack – see chapter 4.6 – investigated to challenge our and other approaches.

Worst case attack (wca): In this attack the colluders try to find out how the fingerprint matrix \mathbf{X} was generated. Therefore they build the so called colluder matrix X_C out of their fingerprints and calculate according probabilities p_C^i for every column i . With these probabilities they try to foresee their accusation score – they calculate *assumed* scores – and generate the manipulated fingerprint y favorably. Finally they compare their assumed accusation scores and further optimize y by changing bits at positions where the colluder fingerprint with highest accusation score and the one with lowest differ from each other. At the end y is re-generated in a way that the accusation scores of all colluders are approximately the same according to their calculated probability p_C^i . This way is proceeded until all of their assumed accusation scores are alike.

Continuing tests: The Monte Carlo estimation is used to verify our tests. Therefore the parameter for $\varepsilon_1 = 10^{-3}$ is chosen as to be comparable to the results of [84] and [74], which at that time were the best so far. Note that recently, *Furon et al.* intensified their work on joint decoding fingerprinting codes and published a series of further improvements to the approach in [74], for instance [73], [75], [40]. The results of our approach compared to [84] and [74] are shown in table 27. One recognizes the shorter code length for every maximum collusion size c_0 . The overall number of colluders caught in different attack scenarios can be seen in Figure 25. We admit that these percentages are slightly lower than those from [74], however, the code complexity is reduced significantly: For the single decoder we worked with lookup tables, so that we only have $m \times n$ summations which is negligible.

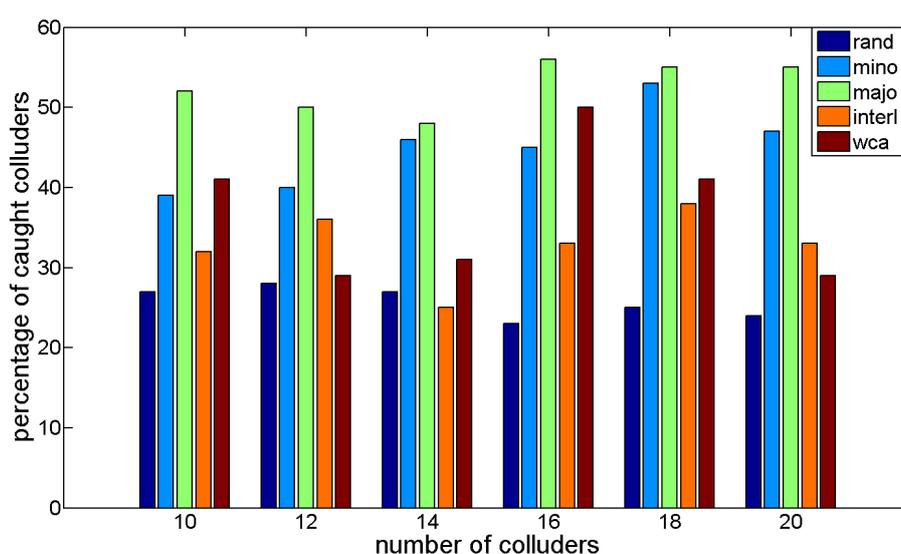


Figure 25: Percentage of colluder fingerprints caught in different attack scenarios

Table 27: Code length comparison for small c_0 with with chosen FP error bound $\epsilon_1 = 10^{-3}$ and 10.000 users

collusion size c_0	<i>Nuida et al.</i> [84]	<i>Meerwald & Furon</i> [74]	Our approach [10]
	single decoder	joint decoder	joint decoder
2	253	~ 304	~ 91
3	877	~ 584	~ 161
4	1454	~ 904	~ 333
6	3640	~ 1616	~ 837
8	6815	~ 2688	~ 1551

Table 28: Experimental results for code length $m = 3671$ with $c_0 = 12$, $\epsilon_1 = 10^{-3}$ and number of users $n = 10^4$

Strategy	Random	Minority vote	Majority vote	Interleaving	Worst Case Attack
False positives	$14 \cdot 10^{-5}$	$16 \cdot 10^{-5}$	$14 \cdot 10^{-5}$	$12 \cdot 10^{-5}$	$18 \cdot 10^{-5}$
False negatives	$34 \cdot 10^{-5}$	$30 \cdot 10^{-5}$	$20 \cdot 10^{-5}$	$38 \cdot 10^{-5}$	$204 \cdot 10^{-5}$

Table 12 shows the code lengths of the scheme presented in chapter 8.1 for $c_0 \geq 10$. To the best of our knowledge there are no joint decoding tracing strategies for these collusion sizes that provide comparably less computational complexity. Because of this, table 12 shows our results compared to the results of [121], which is the algorithm described in chapter 4.5, that we already used for the single decoding step within our joint decoding approach. The table shows the enormous reduction of the code length for the joint decoding strategy compared to the single decoding in [121]. This means, the code length is reduced significantly, compared to an acceptable increment of the effort for fixed error rates.

Tables 13 and 28 show the false positive and false negative errors for collusion sizes of $c_0 = c = 10$ and $c_0 = c = 12$. The experimental results yield error rates that are below the theoretical range for which the parameters were chosen as $\epsilon_1 = 10^{-3}$ and $\epsilon_2 = 1/2$. Note: For reason of comparison the false positive error rate was set to $\epsilon_1 = 10^{-3}$, instead of $n \leq \epsilon_1^{-1}$ as commonly proposed in this work.

In total, this means our scheme provides low false alarm and almost zero no accusation rates while the total number of identified colluders is about 25 – 55% depending on the tested collusion strategies. A significant improvement, however, is the short code length with which these error rates are achieved not requiring any information of the collusion size and attack strategy.

I Regarding the correlative column generation II

This section contains a broader evaluation of the enhanced fingerprint generation process for continuous distributions presented in chapter 8.2. Furthermore we will give the statistical explanations regarding the rank correlation coefficients. The following computations and explanations are an excerpt of the elaborations in [71].

Example of the columnwise generation of a correlated fingerprint matrix

The exemplary – and therefore very small – fingerprint matrix \mathbf{X}' is correlated.

$$\begin{pmatrix} 1 & 1 & \mathbf{1} & 0 & 1 & 0 & 1 & \mathbf{1} \\ 1 & 1 & 0 & 0 & \mathbf{0} & 0 & \mathbf{0} & 0 \\ 1 & \mathbf{0} & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & \mathbf{1} & 1 & \mathbf{1} \\ 1 & 1 & 0 & \mathbf{1} & 1 & 0 & \mathbf{0} & 0 \\ \mathbf{0} & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \mathbf{0} & \mathbf{1} & 1 & 0 \\ 1 & 1 & 0 & \mathbf{1} & 1 & 0 & 1 & \mathbf{1} \\ 1 & 1 & \mathbf{1} & 0 & 1 & 0 & \mathbf{0} & 0 \\ 1 & 1 & 0 & 0 & 1 & \mathbf{1} & 1 & \mathbf{1} \end{pmatrix}$$

The bold (and blue) entries represent those, that have been placed randomly in admitted positions of the corresponding column. The creation of the matrix is depicted below. The gray bars denote rows that are locked for further *significant positions*. We set $\lambda = 0.33$ as correlation bound with an epsilon-environment $\mathcal{B}_\epsilon(\lambda)$ of size $\epsilon = 0.01$ and the ranked probability vector $\hat{p} = (0.90, 0.88, 0.17, 0.19, 0.75, 0.28, 0.70, 0.34)$.

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \mathbf{0} \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 1 & \mathbf{0} \\ 1 & 1 \\ 1 & 1 \\ \mathbf{0} & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & \mathbf{1} \\ 1 & 1 & 0 \\ 1 & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ \mathbf{0} & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & \mathbf{1} \\ 1 & 1 & 0 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & 1 & \mathbf{1} & \mathbf{0} \\ 1 & 1 & 0 & 0 \\ 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & \mathbf{1} \\ 1 & 1 & \mathbf{1} & \mathbf{0} \\ 1 & 1 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 1 & 0 & 0 & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} & 0 & \mathbf{1} \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & \mathbf{1} \\ 1 & 1 & 0 & 0 & \mathbf{0} \\ 1 & 1 & 0 & \mathbf{1} & \mathbf{1} \\ 1 & 1 & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ 1 & 1 & 0 & 0 & \mathbf{1} \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \mathbf{0} & 0 \\ 1 & \mathbf{0} & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & \mathbf{1} \\ 1 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{0} & \mathbf{1} & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \mathbf{0} & \mathbf{1} \\ 1 & 1 & 0 & \mathbf{1} & \mathbf{1} & 0 \\ 1 & 1 & \mathbf{1} & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & \mathbf{1} \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & \blacksquare \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & \blacksquare \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & \blacksquare \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & \blacksquare \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & \blacksquare \\ \mathbf{0} & 1 & 0 & 0 & 1 & 0 & 1 & \blacksquare \\ 1 & 1 & 0 & 0 & \mathbf{0} & 1 & 1 & \blacksquare \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & \blacksquare \\ 1 & 1 & 1 & 0 & 1 & 0 & \mathbf{0} & \blacksquare \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & \blacksquare \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \mathbf{0} & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & \mathbf{0} & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & \mathbf{0} & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Once the matrix is completely correlated, as a last step of the algorithm, for reasons of unpredictability, we suggest to permute the columns to finally receive the fingerprint matrix \mathbf{X} .

The choice of the cutoff parameter t

Until now we have not considered the choice of the cutoff t that bounds the limits for values of the probability vector p , see chapter 4.5. In the following we discuss the conditions required for the choice of an optimal t .

As proven in [121], the cutoff needs to satisfy $c_0 t \ll 1$. As described in chapter 4.5, the probabilities for the column generation of \mathbf{X} prescribe the probability for a symbol '1' in the respective column. It must hold $p_i \in [t, 1 - t]$. Within the algorithm presented in chapter 8.2 into each column i is spread exactly $\lceil p_i \cdot n \rceil$ '1's.

A reasonable lower bound for t is $\frac{1}{n}$. This is because it enforces at least one symbol '1' in every column ($tn = \frac{1}{n}n = 1$). On the other side, because of $(1 - t)n = n - tn = n - \frac{1}{n}n = n - 1$, it enforces at least one symbol '0' as well. This prevents from columns carrying all the same symbol, which would be simply no information at all for the distributor.

All p_i lie within the interval $[t, 1 - t]$. Hence, it must hold $t \leq \frac{1}{2}$. In order to best experience the advantage of a bias distribution function (whether discrete or continuous, see chapter 4.5) the values in p should show a large variation. For this reason the interval $[t, 1 - t]$ should be as large as possible.

To conclude, for a reasonable choice of the cutoff parameter t it should hold:

$$\frac{1}{n} \leq t \ll \frac{1}{2} \quad \text{for } n > 2$$

A broader analysis of the choices for the cutoff can be found in [121] and recently [55] and [62].

Analysis of the correlation

To analyze the correlation we inserted into the fingerprint matrix, we calculate the correlation matrix

$$\mathcal{C} = (\tau_{B_{i,j}}) = \begin{pmatrix} \tau_{B_{1,1}} & \tau_{B_{1,2}} & \cdots & \tau_{B_{1,m}} \\ \tau_{B_{2,1}} & \tau_{B_{2,2}} & \cdots & \tau_{B_{2,m}} \\ \vdots & \vdots & \ddots & \vdots \\ \tau_{B_{m,1}} & \tau_{B_{m,1}} & \cdots & \tau_{B_{m,m}} \end{pmatrix} \in [-1, 1]^{m \times m}$$

where the parameter $\tau_{B_{i,j}}$ denotes the rank correlation coefficient according to *Kendall*, see for instance [47]:

Definition 27 *Be X, Y vectors of the same size with realizations of random variables. Then Kendall's rank correlation coefficient τ_B is defined as*

$$\tau_B = \frac{N_s - N_d}{\sqrt{(N_s + N_d + T_X)(N_s + N_d + T_Y)}},$$

and it holds $-1 \leq \tau_B \leq 1$, where

- N_s denotes the number of concordant pairs in X and Y ,
- N_d denotes the number of different pairs in X and Y ,
- T_X denotes the number of compounds in X ,
- T_Y denotes the number of compounds in Y .

Kendall's τ_B is a measurement for the correlation between two columns i and j . A more efficient method for the calculation of τ_B with $\mathcal{O}(n \log n)$ is given in [60].

Because of its symmetry and because the correlation of a column to itself is always 1, is sufficient to calculate the upper triangular matrix of \mathcal{C} [47].

$$\mathcal{C}' = \begin{pmatrix} 0 & \tau_{B_{1,2}} & \cdots & \tau_{B_{1,m}} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \tau_{B_{m-1,m}} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$

As an upper triangular matrix contains at most $\frac{m(m+1)}{2}$ elements that are not zero, the number of relevant elements in \mathcal{C}' is exactly $m' := \frac{m(m+1)}{2} - m = \frac{m(m-1)}{2}$. Note that the principal diagonal solely contains zeros as well. We summarize the coefficients of the upper triangular matrix \mathcal{C}' , i.e. the entries for which holds $\{\tau_{B_{i,j}} : j > i\}$, in a set

$$\bar{\mathbf{c}} = \{\bar{c}_1, \bar{c}_2, \dots, \bar{c}_{m'}\} \quad (21)$$

Analysis of the correlation coefficients of correlated fingerprinting codes

To evaluate the impact of the correlation, we tested the eleven correlation bounds of Λ

$$\Lambda = \{t, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50\} \quad (22)$$

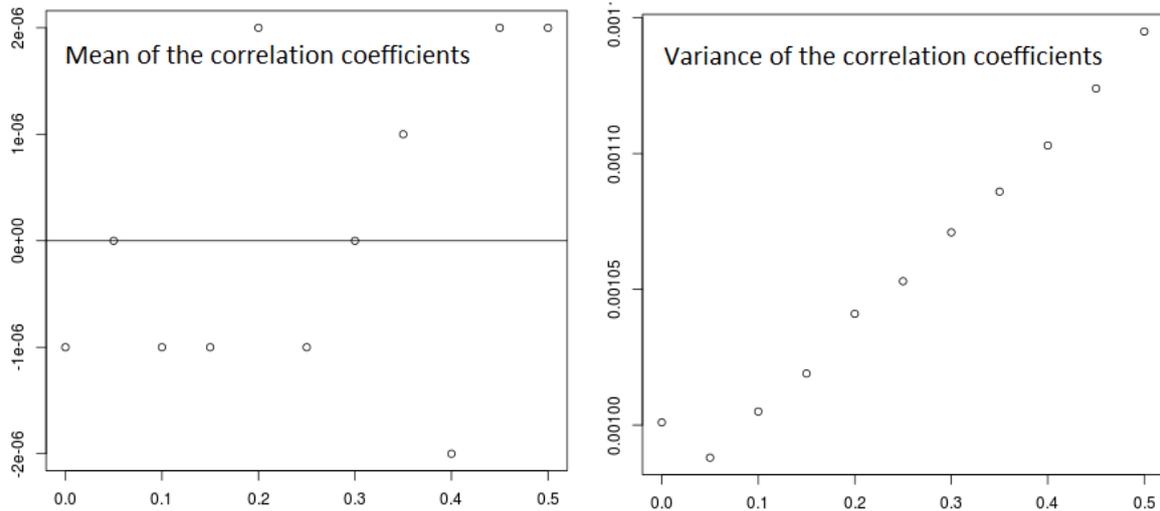


Figure 26: Analysis of the correlation coefficients: arithmetic mean and empirical variance. The x-axis marks the correlation bound λ from 0 to 0.5. Based on [71]

against a set of 500 fingerprint matrices with exemplary parameters $n = 1000$, $\varepsilon_1 = \frac{1}{n}$, $\varepsilon_2 = \frac{1}{2}$, $t = \frac{1}{n}$ and $c_0 = 7$. Because the ranked probability vector \hat{p} is sorted, see chapter 8.2, a correlation of at most 0.50 is sufficient. The choice of the parameters is rather insignificant, as it has no effect on the correlation between the columns. Hence our choice is a very classical parameter set.

For every fingerprint matrix \mathbf{X} , the corresponding set of correlation coefficients \bar{c} is calculated according to equation 21. Next the mean, the variance, the skewness and the excess are determined, and each average of the results for the 500 fingerprint matrices is calculated.

The results can be seen in figures 26 and 27. With increasing correlation, the mean and the skewness are dispersed close to zero, whereas the variance and the excess increase as well. The consequence is a broader spreading of the coefficients and its distribution is leptokurtic. The reason for this an increment of the number of outliers, i.e. an increment of the number of coefficients that are significant far away of zero.

In summary, with a correlation bound of $\lambda = 0.10$ the resulting correlation of a fingerprint matrix \mathbf{X} is hardly recognizable or verifiable. A broader explanation is to be found in [71]

Analysis of the impact of the Interleaving attack on correlated fingerprinting codes

A further test set of 50 fingerprinting codes with parameter sets of $n = 1000$, $\varepsilon_1 = \frac{1}{n}$, $\varepsilon_2 = \frac{1}{2}$, $t = \frac{1}{n}$ und $c_0 = 21$ are tested against 100 runs of Interleaving attacks. The same correlation bound as in equation 22 was chosen. The results are depicted in figures 28 and 29.

What we see is that with a correlation bound of $\lambda = 0.10$, the accusation of innocent-fingerprints reduces by a factor of six, and at the same time the number of colluders caught is slightly higher compared to a correlation bound of $\lambda = 0.0$ (original Tardos). In other words, the false negative error is significantly reduced and at the same time, the false positive error is slightly reduced. The figures also suggests, that results will no more improve significantly with correlation bound of $\lambda = 0.20$ or larger.

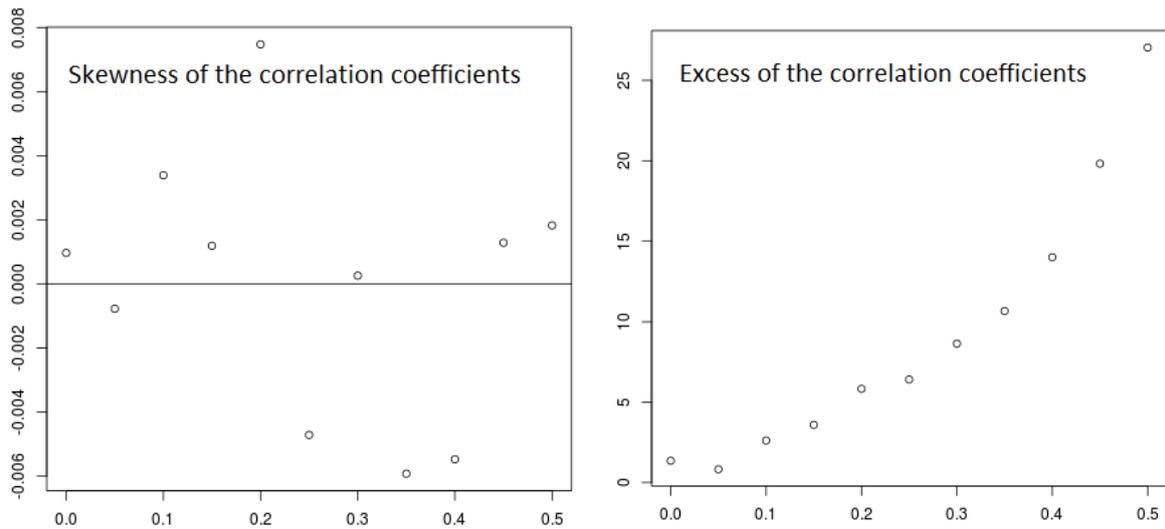


Figure 27: Analysis of the correlation coefficients: skewness and excess. The x-axis marks the correlation bound λ from 0 to 0.5. Based on [71]

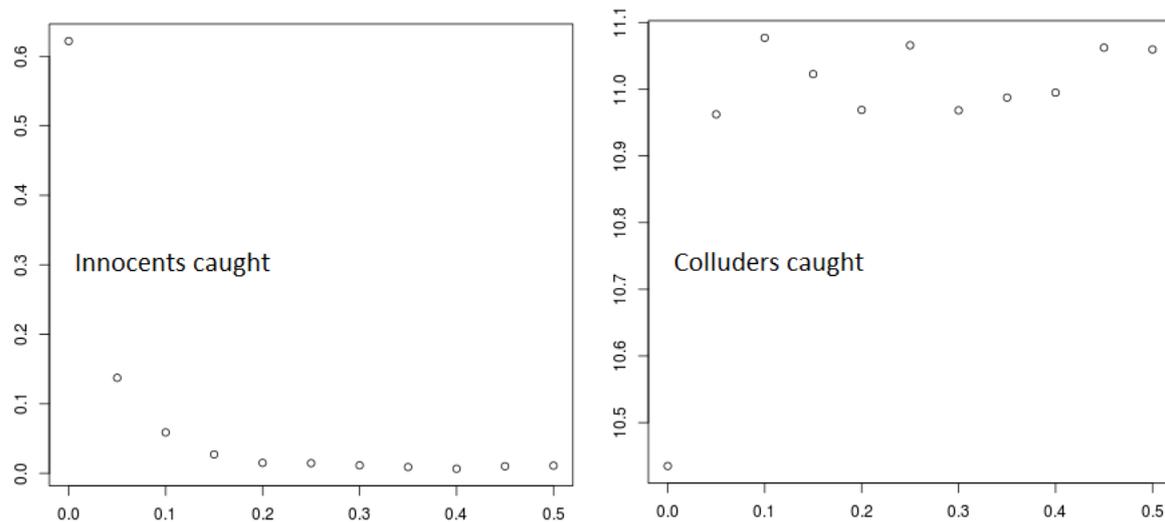


Figure 28: Interleaving attacks on correlated fingerprinting codes: Average number of innocent-fingerprints that are accused (left) and average number of colluders caught (right). Based on [71]

In the same test set, we considered the standard deviation of the scores of innocent-fingerprints after Interleaving attacks. What we see is the expected: For an un-correlated fingerprint matrix \mathbf{X} (i.e. $\lambda = 0.0$), the standard deviation σ_{inn} equals one. This mirrors the pre-conditions on the construction of the Tardos codes [114], see chapter 4.5. Theoretical proofs are given in [121] for instance. With increasing correlation, the standard deviation of the scores of innocent-fingerprints decreases. This effect has been largely discussed in chapters 7.2, 7.3 and in [11].

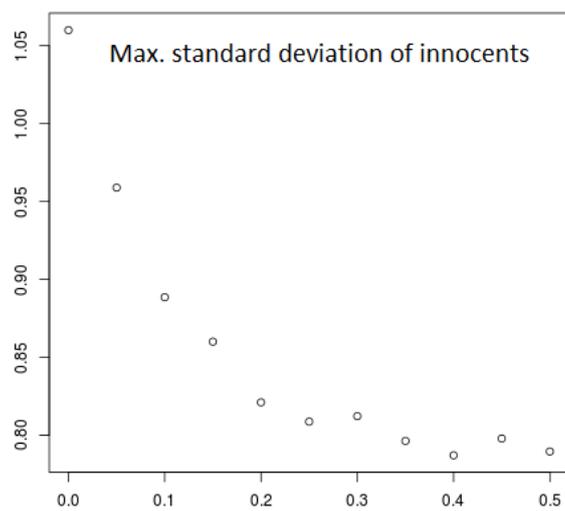


Figure 29: Interleaving attacks on correlated fingerprinting codes: Maximum standard deviation of the accusation scores of innocent-fingerprints. Based on [71]

J Regarding the universal threshold calculation

This section gives derivations and widens the explanations and tests of the universal threshold calculation via mixture models presented in chapter 8.3.

J.1 Derivations of the parameter calculations

The iterative expectation maximization algorithm (EM) is approved to find the maximum-likelihood-estimator for a parameter set of a mixture model, even if the parameter set is not completely known. The incomplete data set here is denoted as $\{S_j\}_{j=1,\dots,n}$, as it represents the accusation scores for the fingerprinting scheme. The unknown information here represents the mapping which component of the mixture model is associated to which accusation score.

As already mentioned in chapter 8.3, the initial parameter set $\Theta^{(0)}$ needs to be selected manually. Afterwards, the following two steps have to be conducted successively with increasing $k \in \mathbb{N}$.

1. **Expectation step:** The actual parameter set $\Theta^{(k)}$ determines the auxiliary function $Q(\Theta^{(k+1)}|\Theta^{(k)})$. All values but $\Theta^{(k+1)}$ are given.
2. **Maximization step:** The function Q is maximized.

$$\Theta^{(k+1)} = \arg \max_{\Theta} Q(\Theta|\Theta^{(k)})$$

Both steps are repeated iteratively until an abort criteria is reached.

Derivation of an EM for a Gaussian mixture model

We will only give a short sketch of the derivation, for more information see [4] and [18]. A more general approach of equation (14) for a Gaussian mixture model with M components as posted in [4] is as follows:

$$p(x|\Theta) = \sum_{j=1}^M \alpha_j \mathcal{N}(x|\mu_j, \sigma_j), \quad \text{with} \quad \sum_{j=1}^M \alpha_j = 1$$

Here the parameter set we want to estimate is denoted as $\Theta = \{\alpha_1, \theta_1, \dots, \alpha_M, \theta_M\} = \{\alpha_1, \mu_1, \sigma_1, \dots, \alpha_M, \mu_M, \sigma_M\}$. According to [4], we can derive the Q -function of the maximization step as follows. For a given data set $\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ the uncomplete data probability is given as

$$\log \mathcal{L}(\Theta|\mathcal{X}) = \log \prod_{i=1}^N \rho(x_i|\Theta) = \sum_{i=1}^N \log \left[\sum_{j=1}^M \alpha_j \mathcal{N}(x_i|\mu_j, \sigma_j) \right].$$

Assuming there exists an unknown data set $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$, with u_i telling which normal distribution $\mathcal{N}(x|\mu, \sigma)$ is responsible for x_i and thereby clearly assigning each x_i to one of the M normal distributions. Also, let us assume that the values for \mathcal{U} are known. Thereby we get the complete data probability as

$$\log \mathcal{L}(\Theta|\mathcal{X}, \mathcal{U}) = \log \rho(\mathcal{X}, \mathcal{U}|\Theta) = \sum_{i=1}^N \log \left(\alpha_{u_i} \mathcal{N}(x_i|\mu_{u_i}, \sigma_{u_i}) \right).$$

Thereby and with the help of *Bayes' theorem*

$$\rho(u_i|x_i, \Theta^{(k)}) = \frac{\alpha_{u_i}^{(k)} \mathcal{N}(x_i|\mu_{u_i}, \sigma_{u_i})}{\rho(x_i|\Theta^{(k)})} = \frac{\alpha_{u_i}^{(k)} \mathcal{N}(x_i|\mu_{u_i}, \sigma_{u_i})}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i|\mu_j, \sigma_j)}$$

we can calculate u_i for given x_i and guessed Θ^k for the whole data set:

$$\rho(\vec{y}|\mathcal{X}, \Theta^{(k)}) = \prod_{i=1}^N p(u_i|x_i, \Theta^{(k)}) = \prod_{i=1}^N \frac{\alpha_{u_i}^{(k)} \mathcal{N}(x_i|\mu_{u_i}, \sigma_{u_i})}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i|\mu_j, \sigma_j)}.$$

The expectation of the Q function can now be derived as

$$Q(\Theta, \Theta^{(k)}) = E[\log \rho(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta']$$

With some complex calculation steps this 'reduces' to

$$Q(\Theta, \Theta^{(k)}) = \sum_{i=1}^N \sum_{l=1}^M \log(\alpha_l \mathcal{N}(x_i|\mu_l, \sigma_l)) p(l|x_i, \Theta^{(k)}) \quad (23)$$

$$= \sum_{i=1}^N \sum_{l=1}^M \log(\alpha_l) \rho(l|x_i, \Theta^{(k)}) + \sum_{i=1}^N \sum_{l=1}^M \log(\mathcal{N}(x_i|\mu_l, \sigma_l)) \rho(l|x_i, \Theta^{(k)}) \quad (24)$$

Now the weight parameters α_l are divided from the parameters θ_l . This allows maximizing them independent of each other.

Finding the weight parameters α_l

To find α_l , we have to calculate the corresponding derivation of $Q(\Theta, \Theta^{(k)})$. As the right sum in equation (24) does not contain α_l we need not consider this term. We insert a *Lagrange multiplier* λ to ensure that the sum over all α_l remains 1.

$$\begin{aligned} \frac{\partial}{\partial \alpha_l} \left\{ \left(\sum_{i=1}^N \sum_{l=1}^M \log(\alpha_l) \rho(l|x_i, \Theta^{(k)}) \right) + \lambda \left(\sum_{l=1}^M \alpha_l - 1 \right) \right\} &\stackrel{!}{=} 0 \\ \Rightarrow \left(\sum_{i=1}^N \frac{1}{\alpha_l} \rho(l|x_i, \Theta^{(k)}) \right) + \lambda &\stackrel{!}{=} 0. \end{aligned}$$

With some more math we receive

$$\alpha_l^{(k+1)} = \frac{1}{N} \sum_{i=1}^N \frac{\alpha_l^{(k)} \mathcal{N}(x_i, \mu_l, \sigma_l)}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i|\mu_j, \sigma_j)}.$$

This is the guide for the calculation of the α_l for equation (14) in chapter 8.3. It calculates and maximizes the expectation value.

Finding the means μ_l

In analogy to α_l , to find the means we have to calculate the corresponding derivations of μ_l of equation (24). Here the left term can be omitted.

$$\frac{\partial}{\partial \mu_l} \left\{ \sum_{i=1}^N \sum_{l=1}^M \log \mathcal{N}(x_i | \mu_l, \sigma_l) \rho(l | x_i, \Theta^{(k)}) \right\} = \frac{1}{\sigma_l^2} \sum_{i=1}^N (x_i - \mu_l) \rho(l | x_i, \Theta^{(k)})$$

Setting this equation to zero gives us

$$\mu_l = \frac{\sum_{i=1}^N x_i \rho(l | x_i, \Theta^{(k)})}{\sum_{i=1}^N \rho(l | x_i, \Theta^{(k)})}$$

Again with the help of *Bayes' theorem*, we yield the formula for $\mu_l^{(k+1)}$.

$$\mu_l^{(k+1)} = \frac{\sum_{i=1}^N x_i \frac{\alpha_l^{(k)} \mathcal{N}(x_i | \mu_l, \sigma_l)}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i | \mu_j, \sigma_j)}}{\sum_{i=1}^N \frac{\alpha_l^{(k)} \mathcal{N}(x_i | \mu_l, \sigma_l)}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i | \mu_j, \sigma_j)}}$$

Finding the standard deviation

We calculate the derivation of $Q(\Theta, \Theta^{(k)})$ for σ_l .

$$\begin{aligned} & \frac{\partial}{\partial \sigma_l} \left\{ \sum_{i=1}^N \sum_{l=1}^M \log \mathcal{N}(x_i | \mu_l, \sigma_l) \rho(l | x_i, \Theta^{(k)}) \right\} \stackrel{!}{=} 0 \\ \Rightarrow & \sum_{i=1}^N \left[-\frac{1}{\sigma_l} + \frac{1}{\sigma_l} \left(\frac{x_i - \mu_l}{\sigma_l} \right)^2 \right] \rho(l | x_i, \Theta^{(k)}) \stackrel{!}{=} 0, \end{aligned}$$

This gives us

$$\sigma_l = \sqrt{\frac{\sum_{i=1}^N (x_i - \mu_l)^2 \rho(l | x_i, \Theta^{(k)})}{\sum_{i=1}^N \rho(l | x_i, \Theta^{(k)})}}$$

and finally with the help of *Bayes' theorem* we receive

$$\sigma_l^{(k+1)} = \sqrt{\frac{\sum_{i=1}^N (x_i - \mu_l^{(k+1)})^2 \frac{\alpha_l^{(k)} \mathcal{N}(x_i | \mu_l, \sigma_l)}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i | \mu_j, \sigma_j)}}{\sum_{i=1}^N \frac{\alpha_l^{(k)} \mathcal{N}(x_i | \mu_l, \sigma_l)}{\sum_{j=1}^M \alpha_j^{(k)} \mathcal{N}(x_i | \mu_j, \sigma_j)}}}$$

allowing to estimate the standard deviation σ_l for step $k + 1$.

J.2 Test setup for the fitting of mixture models and backup threshold if the fitting fails

We estimate the initial parameter set $\Theta^{(0)} = \{\alpha_1, \mu_1, \sigma_1, \alpha_2, \mu_2, \sigma_2\}$ for a Gaussian mixture model as follows.

$$\begin{array}{lll} \alpha_1 = \frac{n-c_0}{n} & \mu_1 = 0 & \sigma_1 = 1 \\ \alpha_2 = \frac{c}{n} & \mu_2 = \frac{1}{2} \left(Z^* + \max_{j \in \{1, 2, \dots, n\}} \{S_j\} \right) & \sigma_2 = 1 \end{array}$$

The initial values for μ_1 and σ_1 are due to the general requirements for the fingerprint generation process according to [114]. We expect the mean μ_2 to lie between Z^* and the maximum accusation score.

Backup threshold if Z^* fails

As described in chapter 8.3 the threshold decision is given so that the output set of the tracing algorithm is as:

$$\mathbf{T} = \{X_j | \forall j = \{1, \dots, n\} : S_j > Z^*\}$$

What is left aside, the fitting of a mixture model may fail. This may happen for example if there are too less accusation scores of colluder fingerprints. In that case both components of the mixture model will be estimated as equal and the value for μ_2 becomes disproportional low. Therefore, for small values for c_0 , we propose to add a backup threshold. This way, if the mixture models fail, we fall back to the threshold proposed by [121] for instance. This means, if it holds $Z^* = \mu_2 < Z$, we employ Z of equation (7) as threshold. Hence, the number of innocent fingerprints output by the tracing algorithm cannot increase and the false positive error remains bounded. However, during all tests with the universal threshold calculation method from chapter 8.3 the fitting never failed. Hence, for collusion sizes $c \leq 13$, see table 15 in chapter 8.3, the empirically calculated probability that the fitting fails is less than 10^{-4} . For smaller collusion sizes, we already suggest other approaches, see chapter 10.

J.3 Tests for the minority and majority vote attack

In chapter 8.3 we only showed results for the interleaving attack (table 15). This section also presents the results for the minority vote attack and majority vote attack.

Tables 29 and 30 carry the results obtained with the same parameter selection as for the results listed in table 15 in chapter 8.3. That is the code length was set to $m = 12,248, 22,912$ and $47,556$ for $c_0 = 15, 21$ and 31 respectively. The error bounds were as $\varepsilon_1 = 1/n$ and $\varepsilon_2 = 1/2$. As mentioned in chapter 8.3, we ran 10,000 attacks for the symmetric score function [121] and for the interleaving score function [86] as well as for the score function proposed in [62] and per selection of c .

The results differ in that, on the one hand, the minority vote attack is the least violent for the GMM, which means this choice of threshold calculation is much more reliable than the ordinary threshold calculation Z^* . On the other hand, against the majority vote attack the GMM provides no significant improvement at all, whereas the ordinary threshold calculation competes very well with this kind of collusion strategy.

Table 29: Results for FP and $|C|$ with different threshold calculations (Z^* , GMM^+ , GMM^-), different score functions and varying collusion sizes c while $c_0 = 15, 21$ and 31 after a minority attack.

c_0	c	Threshold	'Škorić' [121]		'Oosterwijk' [86]		'Laarhoven' [62]	
			FP	$ C $	FP	$ C $	FP	$ C $
$c_0 = 15$	$c = 13$	Z^*	0.26	12.75	0.00	12.35	0.00	4.02
		GMM^-	0.27	12.74	1.78	13.00	0.36	12.90
		GMM^+	0.00	6.34	0.00	7.01	0.00	6.46
	$c = 15$	Z^*	0.25	13.49	0.00	12.91	0.00	1.69
		GMM^-	0.26	13.48	1.79	14.99	0.36	14.34
		GMM^+	0.00	7.07	0.00	8.30	0.00	7.34
	$c = 18$	Z^*	0.25	12.21	0.00	12.17	0.00	0.39
		GMM^-	0.25	12.11	1.79	17.92	0.36	14.89
		GMM^+	0.04	7.72	0.00	10.40	0.01	8.47
$c_0 = 21$	$c = 18$	Z^*	0.26	17.50	0.00	17.71	0.00	8.67
		GMM^-	0.27	17.50	1.09	18.00	0.34	17.95
		GMM^+	0.00	8.80	0.00	8.87	0.00	8.98
	$c = 21$	Z^*	0.26	18.20	0.00	19.49	0.00	4.11
		GMM^-	0.27	18.17	1.08	21.00	0.34	20.52
		GMM^+	0.00	9.89	0.00	10.49	0.00	10.39
	$c = 24$	Z^*	0.25	16.66	0.00	19.73	0.00	1.64
		GMM^-	0.25	16.52	1.09	23.98	0.34	22.03
		GMM^+	0.04	10.68	0.00	12.22	0.00	11.67
$c_0 = 31$	$c = 24$	Z^*	0.26	23.71	0.00	23.99	0.00	20.70
		GMM^-	0.27	23.71	0.71	24.00	0.32	24.00
		GMM^+	0.00	11.90	0.00	11.66	0.00	11.94
	$c = 31$	Z^*	0.27	25.28	0.00	30.15	0.00	10.51
		GMM^-	0.27	25.20	0.71	31.00	0.34	30.72
		GMM^+	0.01	14.64	0.00	15.15	0.00	15.40
	$c = 38$	Z^*	0.27	19.79	0.00	32.03	0.00	2.93
		GMM^-	0.24	19.18	0.72	37.97	0.34	34.94
		GMM^+	0.15	16.11	0.00	18.77	0.00	18.59

The reason for this is by design of the attacks. In the minority vote attack, the attackers have a higher potential to stay below the ordinary threshold Z^* , because the number of amounts added to their score is the least possible (*minority vote*). As for the GMM, the threshold is dynamic with the accusation scores and therefore adjusted for each attack. Hence, the chance for the attackers to create a fingerprint for which their accusation scores stay below the threshold, reduces decisively. The opposite is the case for the majority vote attack. In this attack the colluder-fingerprints' accusation scores consist of a significant large number of positive amounts (*majority vote*), for which already calculating the Hamming Distance as accusation

Table 30: Results for FP and $|C|$ with different threshold calculations (Z^* , GMM^+ , GMM^-), different score functions and varying collusion sizes c while $c_0 = 15, 21$ and 31 after a majority attack.

c_0	c	Threshold	'Škorić' [121]		'Oosterwijk' [86]		'Laarhoven' [62]	
			FP	$ C $	FP	$ C $	FP	$ C $
$c_0 = 15$	$c = 13$	Z^*	0.21	12.81	0.00	12.97	0.00	12.95
		GMM^-	0.21	12.80	0.26	13.00	0.26	13.00
		GMM^+	0.00	6.36	0.00	6.50	0.00	6.51
	$c = 15$	Z^*	0.20	13.87	0.00	14.08	0.00	13.81
		GMM^-	0.21	13.86	0.25	15.00	0.25	15.00
		GMM^+	0.00	7.04	0.00	7.50	0.00	7.50
	$c = 18$	Z^*	0.20	13.29	0.00	9.90	0.00	9.08
		GMM^-	0.20	13.19	0.26	18.00	0.26	18.00
		GMM^+	0.01	7.72	0.00	9.02	0.00	9.01
$c_0 = 21$	$c = 18$	Z^*	0.24	17.68	0.00	17.97	0.00	17.96
		GMM^-	0.24	17.68	0.27	18.00	0.27	18.00
		GMM^+	0.00	8.85	0.00	9.02	0.00	9.02
	$c = 21$	Z^*	0.23	19.07	0.00	19.86	0.00	19.62
		GMM^-	0.23	19.05	0.26	21.00	0.26	21.00
		GMM^+	0.00	9.96	0.00	10.50	0.00	10.50
	$c = 24$	Z^*	0.22	18.43	0.00	16.89	0.00	16.18
		GMM^-	0.22	18.32	0.26	24.00	0.26	24.00
		GMM^+	0.01	10.79	0.00	12.00	0.00	12.00
$c_0 = 31$	$c = 24$	Z^*	0.24	23.87	0.00	24.00	0.00	24.00
		GMM^-	0.24	23.87	0.26	24.00	0.26	24.00
		GMM^+	0.00	11.92	0.00	12.04	0.00	12.04
	$c = 31$	Z^*	0.23	27.53	0.00	29.43	0.00	29.22
		GMM^-	0.23	27.47	0.25	31.00	0.25	31.00
		GMM^+	0.00	14.82	0.00	15.53	0.00	15.53
	$c = 38$	Z^*	0.23	24.27	0.00	20.27	0.00	19.48
		GMM^-	0.22	23.84	0.27	37.99	0.26	37.99
		GMM^+	0.05	16.86	0.00	19.02	0.00	19.01

score oftentimes suffices to catch the colluders. For this reason the colluders' scores have very high probability to stay above the ordinary threshold Z^* . This means there hardly is room for improvement. Consequently, the dynamic threshold via GMM cannot improve those results. Note that for the symmetric score function in [121] – by construction – the attack strategy has only marginal influence on the scores for which the results via GMM are very alike the ordinary threshold.

Table 31: Results for correlated and uncorrelated \mathbf{X} with $n = 1.000$ and symmetric threshold calculation [121] compared to GMM for different attacks.

Attack strategy	$c_0 = c$	correlation bound λ	Symmetric Tardos Z		GMM	
			FP	$ C $	FP	$ C $
Interleaving	21	0.00	0.622	10.435	0.26	6.99
		0.10	0.057	11.065	0.03	8.77
	31	0.00	0.576	15.744	0.264	11.76
		0.10	0.040	15.796	0.019	13.39
Minority vote	21	0.00	0.677	11.833	0.23	8.10
		0.10	0.391	10.436	0.22	8.17
	31	0.00	0.622	14.802	0.364	11.67
		0.10	0.294	14.301	0.221	12.63
Majority vote	21	0.00	0.639	10.423	0.26	7.22
		0.10	0.024	11.217	0.01	8.93
	31	0.00	0.618	15.438	0.265	11.10
		0.10	0.101	15.960	0.067	13.54

J.4 Combination of correlated matrix generation and GMM

We also tested the combination of the threshold calculation via Gaussian mixture models (GMM) and the correlative fingerprint matrix generation we introduced in chapter 8.2. For this test set each attack was run 100 times for each 10 different fingerprint matrices \mathbf{X} . All matrices were generated according to the original Tardos scheme [114]. Further parameter choices were $n = 1000$, $c_0 = \{21, 31\}$, $\varepsilon_1 = \frac{1}{n}$, $\varepsilon_2 = \frac{1}{2}$ and $t = \frac{1}{n}$. In table 31 the results for the symmetric score function and the corresponding threshold Z according to [121] as described in chapter 4.5 are depicted.

Besides the average number of colluder-fingerprints output by the tracing algorithm $|C|$, we again listed the average number of misleadingly output innocent-fingerprints. It is referred to as FP according to the false positive error.

The fingerprints were generated according as in the original Tardos scheme [114], i.e. with the original parameter selection instead of the optimal parameter selection as described in appendix A. This is because the original generation forms the basis for the approach with the correlated fingerprint matrix and to be comparable to the corresponding results presented in chapter 8.2. For table 31, we selected the initial parameters for the GMM according to section J.2.

We observe a significant decrement of the average number of innocent fingerprints output by the tracing algorithm for the mixture model approach compared to the symmetric Tardos scheme. The combination using the correlative matrix generation approach from chapter 8.2 and mixture models further reduce the occurring errors. Note that the results in table 31 slightly differ from the equivalent results in table 15 of chapter 8.3 and tables 29 and 30 in appendix J.3. This is because of the different parameter selection mentioned above.

Akademischer Werdegang:

Hochschulbildung:

seit 01/2011	Wissenschaftlicher Mitarbeiter, Fraunhofer SIT, Darmstadt Promovent der TU Darmstadt
10/2001 - 09/2010	Bergische Universität Wuppertal Diplomstudiengang Mathematik, Abschluss: Diplom

Schulausbildung:

08/1992 - 06/2001	Gymnasium August-Dicke-Schule, Solingen Abschluss: Abitur
-------------------	--