Collusion-resistant traitor tracing schemes

by T.M.M. Laarhoven

June 2011

Acknowledgements

This report is the result of many months of hard, but also very enjoyable work at Irdeto and at the TU/e. Before continuing with the content of this report I would like to take a moment to thank all those without whom I could not have written this report.

First of all, I am very grateful to Benne de Weger and Jeroen Doumen (and the TU/e and Irdeto) for making this project possible, and for guiding me throughout the project, both from the TU/e and from Irdeto. If I had any questions or problems, they were always there to help and guide me. I would also like to thank both for letting me choose the further directions I wanted to take with the project. This way I could investigate those things which I thought were most interesting at the time, even though sometimes this led to nothing.

Secondly, I would like to thank Boris Skoric and Peter Roelse for helping me in many ways. Their door was also always open for me, and I have had many useful discussions with both. In this sense I had the great benefit of having two supervisors both at the TU/e (Benne and Boris) and at Irdeto (Jeroen and Peter). Thus if one of them was not available (business trip, appointments, teaching), I could always go see the other, allowing me to address any problems I had quickly, both at the TU/e and at Irdeto.

For discussions on the topic, I would further like to thank the members of the CREST-group at the TU/e. The weekly meetings on Friday were always interesting and provided me with a lot of useful insights, not to mention that I also very much enjoyed the weekly meetings.

Last but not least, I would like to thank my family, my fellow students at the TU/e, and the employees of Irdeto in Eindhoven for creating such a good atmosphere at home, at the TU/e and at Irdeto. I really enjoyed working on the project and writing this report, thanks to this good atmosphere, and I hope the reader will enjoy reading the report as well.

Thijs Laarhoven,

June 2011

Contents

Chapter 1

Introduction

These days many digital products, such as videos and software, are produced and sold. Like most products, these digital products are copyrighted, and distributors demand that people who want to use these products should buy them from authorized resellers. However, with the technology available nowadays it is easy for malicious customers (**pirates** or **traitors**) to produce identical copies of their copy of the digital data, e.g. by copying DVDs, and distribute these copies among others. This then allows people who did not pay for the product to use it anyway, and it allows pirates to make money from this piracy. As an example, one can think of movies being uploaded to YouTube, software being distributed over the internet through torrents, or pirates publishing data needed for getting access to pay-TV channels. An obvious solution to this problem is to add unique **watermarks** or **fingerprints** to each copy, so that if a pirate distributes his watermarked copy and the distributor detects this illegal copy online, the copyright owners can see that the copy belongs to that specific customer, and take steps against that person.

			w			w	w				w		w	w	w		
Alice	0	1	1	1	0	0	1	1	1	0	1	1	0	1	0	0	...
Bob	0	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	...
Charlie	0	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	...
Dave	0	1	1	1	0	0	0	1	1	0	1	1	0	0	0	0	...
Eve	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	0	...
Forgery	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	0	...

Table 1.1: *Watermarked data for five users, and an illegally published copy of the data. Columns marked with a "w" correspond to positions of the watermark, while the rest of the data is the same for every user. Since the watermark of the published copy matches the watermark of Eve, the distributor can easily establish that Eve must be the one having distributed her copy.*

However, this is not the end of the story. Assuming that the actual digital data of the products the people receive are identical, the only differences between two fingerprinted copies are differences in their fingerprints. So if changing the fingerprint does not render the product unusable, a **coalition** (or **collusion**) of multiple pirates (**colluders**) can compare their differently fingerprinted products and detect positions where their products differ, which must thus be part of the fingerprints. So instead of distributing one of their copies, which would implicate one of them, a coalition can create a new copy with a different fingerprint. This new copy then has the same digital data, but a fingerprint that could be a combination of the fingerprints of the colluders. Then tracing guilty users suddenly becomes a non-trivial problem.

			w			w	w				w		w	w	w		
Alice	0	1	1	1	0	0	1	1	1	0	1	1	0	1	0	0	...
Bob	0	1	1	1	0	1	0	1	1	0	1	1	1	1	1	0	...
Charlie	0	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	...
Dave	0	1	1	1	0	0	0	1	1	0	1	1	0	0	0	0	...
Eve	0	1	0	1	0	1	0	1	1	0	1	1	1	0	0	0	...
Forgery	0	1	1	1	0	0	0	1	1	0	1	1	0	1	0	0	...

Table 1.2: *The same watermarked data for the same five users, but with a different forged copy of the data. Since the watermark of the forgery does not match the watermark of any of the users, the distributor cannot easily establish who colluded to create this forged copy. In this case Bob, Dave and Eve colluded to form the forged copy, but one can check that Alice and Charlie (who are innocent) together could also have created the forgery.*

The reason why this problem with colluders is hard to solve can be seen from the incompatible requirements on the fingerprints. On the one hand the distributor wants to be able to distinguish between different users effectively, so the fingerprints should not be too similar to eachother. This means that different fingerprints assigned to different users should also contain many differences. But on the other hand, if the fingerprints assigned to the pirates are very different, then by comparing their copies these colluders can detect and edit a big part of the fingerprints. So for that reason we would rather want pirates to have similar fingerprints with many identical symbols. This gives a contradiction, as fingerprints cannot be both very different and very similar.

Fortunately, there are mathematicians who investigated this problem and came up with so-called **fingerprinting schemes** to solve this problem. Using fingerprinting codes and an appropriate accusation algorithm, it is possible to identify the traitors, even if the fingerprint discovered in the illegally redistributed data does not match any of the colluders' fingerprints.

1.1 Outline

The outline of the report is as follows. First, in Chapter 2, we introduce some terminology related to the problem sketched above, and try to formulate the problem in a clear, mathematical way. In Chapter 3 we then summarize some results from several areas of mathematics which appear when trying to solve this problem. Then in Part I, which contains Chapters 4, 5, 6 and 7, we thoroughly investigate several schemes from literature. The four chapters in this part correspond to the four main models described in Chapter 2. Part I is roughly based on work done in the first half of the project.

Next, in Part II, containing Chapters 8, 9, 10 and 11, we present some new results, which improve upon earlier results found in literature. Basically there are three results which we discuss here: (i) combining the Blayer-Tassa analysis of the Tardos scheme with the analysis done by Skoric et al. (Chapter 8); (ii) using the Tardos scheme (with the improvements from (i)) in the dynamic model, such that with roughly the same codelength as the static Tardos scheme we can catch *all* pirates (Chapter 9); and (iii) improving upon the result described in (ii) by removing the dependency on c, such that we obtain a fully dynamic Tardos scheme (Chapters 10 and 11). The final result (iii) drastically improves upon earlier results from Tassa [Tas05], by achieving a codelength which is approximately the square root of the codelength of Tassa.

After that, in Chapter 12 we briefly discuss publications resulting from this project, besides this final report. These are still a work in progress, and only for one of those publications we give a

preliminary version in Appendix A. Finally, in Chapter 13, we summarize the results from this report, and we mention areas for further research.

Chapter 2

Terminology

As was mentioned in the Introduction, the problem of fighting one pirate is quite easy to solve, as one can simply generate unique watermarks for each copy and embed them (hide them) in the content. For fighting multiple pirates, there is no simple solution, and many suggestions have been made to solve this problem efficiently. These solutions are very mathematical in nature, and as one may know, mathematicians do not like wasting time and paper by using too many words. Therefore mathematical reports generally contain many symbols and abbreviations (e.g. i.e.), and this report is no exception. The disadvantage however is that we first have to explain all the notations, which we will try to do here.

2.1 Fingerprinting codes

First of all, let us formulate the problem with pirates, colluders and fingerprints in a more abstract and mathematical way. For this we use some realistic assumptions. For example, in the most basic form, digital data can be represented as a long string of symbols, most often bits. To make no specific assumptions yet, we can safely assume that data can be represented by a long finite string of symbols from some finite **alphabet** Q of some size q. We often assume without loss of generality that $Q = \{0, 1, \ldots, q-1\}$. The case $q = 2$ then reduces to the **binary alphabet** $Q = \{0, 1\}$. Similar to the case of one pirate and using watermarks, in the case of multiple pirates we also assume that to trace traitors, we add **watermarks** or **fingerprints** to the data. These fingerprints are then finite strings of some **codelength** ℓ over Q.

Using these fingerprints, one can construct a **fingerprinting code** by assigning to each **user**, i.e. each participant in the scheme, a unique fingerprint. We call these fingerprints **codewords**, and we work with them as vectors from Q^ℓ. We usually denote the codeword assigned to user j by $\vec{x}_j$. Writing n for the total number of users in the system, a fingerprinting code is thus a collection of n vectors over Q of length ℓ. We often write $\mathcal{C}$ to denote this fingerprinting code, e.g. $\mathcal{C} = \{\vec{x}_1, \ldots, \vec{x}_n\}$. Also, for convenience later on, we associate to a fingerprinting code a **fingerprinting code matrix** X which has n rows and ℓ columns, and entries $X_{ji} = (\vec{x}_j)_i$, where $(\vec{x}_j)_i$ is the symbol on position i of the codeword $\vec{x}_j$.

Note that all these codewords $\vec{x}_j$ consist solely of the fingerprints. In this abstraction we completely disregard the original product data and the embedding of the fingerprint in the data. Also remark that when we talk about codewords, we really mean elements from $\mathcal{C}$; any string of ℓ symbols from Q that is not a codeword is simply referred to as a **word**.

2.1.1 Coalitions and forgeries

We now turn our attention to the pirates. We usually write C for a **coalition** of pirates, and its size is usually denoted by c. So $C = \{j_1, \ldots, j_c\}$ for some $j_1, \ldots, j_c$, so that this coalition possesses the codewords $\vec{x}_{j_1}, \ldots, \vec{x}_{j_c}$. Using these codewords, a coalition is able to produce a **forgery**, i.e. some pirated fingerprint which is embedded in the data they distribute. A forgery is usually denoted by $\vec{y}$, and it may well be that $\vec{y} \notin \mathcal{C}$, i.e. $\vec{y}$ is a word but possibly not a codeword. For describing which forgeries a coalition can generate, we define the **span** (also known as the **feasible set** or **envelope**), written as $\langle \vec{x}_{j_1}, \ldots, \vec{x}_{j_c} \rangle$, as the set of all forgeries $\vec{y}$ that can be generated by a coalition possessing the codewords $\vec{x}_{j_1}, \ldots, \vec{x}_{j_c}$. The exact definition of this span depends on the scenario, but we do make one universal assumption about the span, namely the **Marking Assumption**: if for some position i we have $(\vec{x}_{j_1})_i = (\vec{x}_{j_2})_i = \ldots = (\vec{x}_{j_c})_i = s$ for some symbol $s \in Q$, i.e. all members of the coalition see the same symbol on position i, then also any forgery $\vec{y}$ generated by this coalition satisfies $(\vec{y})_i = s$. In other words, if a coalition cannot **detect** a fingerprint position (by noticing a difference in their symbols), then they cannot change the symbol. This assumption is motivated by the fact that after the embedding of the fingerprint, it is (assumed to be) impossible for pirates to distinguish between fingerprint positions and positions of the digital data. Thus if pirates would change data on undetectable positions, they would risk editing the product data which could possibly render the product unusable.[1]

2.1.2 Attack models

Besides the Marking Assumption, which is an assumption used in all scenarios, there may be different assumptions on the definition of the span. Let us fix some position i, and let $S = \{s_1, \ldots, s_c\}$ be the c (not necessarily different) symbols seen by a coalition on some position i. Assume that not all symbols in S are the same, so that the Marking Assumption does not hold for this position. Then there are some common **attack models** for determining which values y_i can have. First of all, the **restricted digit model** says that $y_i \in S$ for all positions i. This may be a reasonable assumption when for instance symbols are implemented as different decryption keys of a segment of data. Then it is reasonable to assume that pirates can only distribute one of their keys, if they want their customers to have a working product. Also, if symbols are really decryption keys, then it makes sense to assume that from a subset of all keys, pirates cannot deduce the other keys. This model is the most restrictive model for pirates, and in some cases it may be too restrictive. Therefore a different model considered sometimes is the **arbitrary digit model**, which says that we only assume that $y_i \in Q$. This may be a more practical assumption when symbols are really just symbols taken from a simple finite alphabet Q. Then pirates may also know this alphabet, and then they do not need to see all symbols on position i to be able to put one of those other symbols there. Finally sometimes the previous two scenarios are considered, but with **erasures** allowed. This means that on detectable positions pirates can introduce the unreadable symbol ?, which is not part of the alphabet. This gives pirates the advantage of not having to output one of their symbols (as is required in the restricted digit model), but a disadvantage of outputting an erasure may be that the distributor can see that the pirates detected this position.

Although the above attack models are really different in most cases, one can prove that if we work with the binary alphabet, all attack models are equivalent. In the binary case we have $q = 2$, so then $|S| > 1$ implies $|S| = 2$ and thus $S = Q$. So then obviously the restricted digit

[1] Some argue that this assumption may be too strong, and therefore some papers investigated the scenario when pirates are allowed to edit a small fraction of all undetectable positions, i.e. when pirates are allowed to add some noise to the data. However, this only complicates matters even more, while the results do not drastically change; the same solution methods then still provide good solutions to the problem. For simplicity we will therefore not investigate this model here.

model and arbitrary digit model are equivalent. As for erasures, a distributor can simply replace all erasures in a forgery by the same symbol (say a 0), which is then also a forgery that could have been generated by the coalition. In other words, this way the distributor can simulate that the pirates used some specific strategy on all these erased position, so that pirates are actually better off not using any erasures. So in the binary case, all attack models are essentially the same and give pirates the same amount of power.

2.1.3 Pirate strategies

Now that we have described what pirates are allowed to do in several attack models, we can also investigate some common **pirate strategies** which pirates may use to create forgeries. Pirate strategies are sometimes also denoted by ρ, so that $\vec{y} = \rho(\vec{x}_{j_1}, \ldots, \vec{x}_{j_c})$. The fingerprinting schemes we will discuss later should of course work against any pirate strategy, as the distributor cannot force pirates to use a certain strategy and may not even know the strategy employed by the coalition. Also, the list of pirate strategies given below is by no means complete, as there are thousands of ways to choose $\vec{y}$ from $\langle \vec{x}_{j_1}, \ldots, \vec{x}_{j_c} \rangle$. We just mention some which are the best, the most often considered or just the most intuitive.

Let $S = \{s_1, \ldots, s_c\}$ again be the collection of symbols seen by the coalition on some position i, and let $\alpha_s = \alpha_s(S)$ denote the number of occurrences of symbol $s \in Q$ in the set S. Then $\sum_{s \in Q} \alpha_s(S) = c$. Also let $a \in_R B$ denote a uniformly random choice of one symbol from B, e.g. the probability that we choose $a = b \in B$ is $\alpha_b(B)/|B|$. Finally let S' be the set S when taken as a strict set, i.e. with all duplicates removed. Then some common pirate strategies are as follows.

1. Always s_0: Fix some symbol s_0 in advance, and let $y_i = s_0$ whenever possible.
2. Random strategy: Let $y_i \in_R S'$, e.g. if $s \in Q$, then the probability that $y_i = s$ is $1/|S'|$.
3. Interleaving attack: Let $y_i \in_R S$, e.g. if $s \in Q$, then the probability that $y_i = s$ is $\alpha_s(S)/|S|$.
4. Minority voting: Let y_i be that symbol $s \in Q$ that occurs the least often in S.
5. Majority voting: Let y_i be that symbol $s \in Q$ that occurs the most often in S.
6. Scapegoat strategy: Fix some user $j \in C$ and let $y_i = (\vec{x}_j)_i$ for all i.

Note that we are a bit sloppy above, as the definitions are incomplete and therefore possibly ambiguous. For example there may not be a unique majority/minority symbol, and in the restricted digit model the symbol s_0 may not be an allowed symbol on position i for a given coalition. Even these borderconditions may have a great impact on the output forgery $\vec{y}$ and whether pirates will get away with their illegal activities or not.

2.1.4 Example

Let us now briefly consider an example of all the above terminology. Suppose we have $n = 3$ users, an alphabet $Q = \{0, 1, 2\}$ of size $q = 3$ and a codelength of $\ell = 4$. Let the fingerprinting code $\mathcal{C}$ be given as follows.

$$\mathcal{C} = \{(0, 1, 2, 2), (1, 0, 2, 0), (2, 1, 0, 0)\}$$

Then the code matrix associated to this code is as follows.

$$X = \begin{pmatrix} 0 & 1 & 2 & 2 \\ 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Alice	0	1	2	2
Bob	1	0	2	0
Charlie	2	1	0	0

Table 2.1: *The codewords from $\mathcal{C}$, assigned to the users Alice, Bob and Charlie.*

Table 2.1 below also shows the distribution of codewords to users. Suppose the first two users form a coalition, i.e. $C = \{1, 2\}$. Then the positions $1, 2, 4$ are detectable, while on position 3 the marking assumption applies. If we are in the restricted digit model, then the span of their codewords is given by the following $2^3 = 8$ words.

$$\langle (0,1,2,2), (1,0,2,0) \rangle_{\text{RD}} = \{(0,0,2,0), (0,0,2,2), (0,1,2,0), (0,1,2,2), \tag{2.1}$$
$$(1,0,2,0), (1,0,2,2), (1,1,2,0), (1,1,2,2)\} \tag{2.2}$$

Similarly, in the arbitrary digit model the span can be calculated as the following set of $3^3 = 27$ words.

$$\langle (0,1,2,2), (1,0,2,0) \rangle_{\text{AD}} = \{(0,0,2,0), (0,0,2,1), (0,0,2,2), (0,1,2,0), (0,1,2,1), \tag{2.3}$$
$$(0,1,2,2), (0,2,2,0), (0,2,2,1), (0,2,2,2), (1,0,2,0), \tag{2.4}$$
$$(1,0,2,1), (1,0,2,2), (1,1,2,0), (1,1,2,1), (1,1,2,2), \tag{2.5}$$
$$(1,2,2,0), (1,2,2,1), (1,2,2,2), (2,0,2,0), (2,0,2,1), \tag{2.6}$$
$$(2,0,2,2), (2,1,2,0), (2,1,2,1), (2,1,2,2), (2,2,2,0), \tag{2.7}$$
$$(2,2,2,1), (2,2,2,2)\}. \tag{2.8}$$

Using the always 1 strategy, with a 2 if it is not possible to output a 1, the forgery would be chosen as $(1, 1, 2, 2)$, while the random strategy may give any of the feasible words with equal probability. If the forgery was indeed $\vec{y} = (1, 1, 2, 2)$ and if we were working in the restricted digit model, then from position 4 of the forgery the distributor could conclude that user 1 must be part of the coalition, since he is the only one with the symbol 2 on that position. Similarly in this case user 2 would be caught because $\vec{y}_1 = (\vec{x}_2)_1 = 1$, while no other users have a 1 there.

2.2 Fingerprinting schemes

Using fingerprinting codes, one can build **fingerprinting schemes**, which are informally defined as the whole scheme to trace pirates. For this a code alone is not sufficient; one also needs a **tracing algorithm** σ, mapping a forgery $\vec{y}$ to a set C' of accused users. Also, a fingerprinting scheme may use several fingerprinting codes instead of just one code, which shows that a fingerprinting scheme is really more than just a fingerprinting code.

We distinguish four types of fingerprinting schemes, based on two properties of a scheme: whether it is deterministic or probabilistic, and whether it is static or dynamic. Below we will explain these concepts.

2.2.1 Deterministic versus probabilistic

One of the choices that has to be made by the distributor is whether the whole process, and in particular the tracing of pirates, is done deterministically or probabilistically. In this context, we define a **deterministic scheme** to be a scheme where the tracing of traitors is purely deterministic; when we claim we have caught a pirate, we can really prove that we have caught

a pirate. This in contrast to **probabilistic schemes**, where these statements are always with a certain **error probability** ϵ. A tracing algorithm is therefore a deterministic tracing algorithm if it really catches at least one pirate and does not accuse any innocent users, while such an algorithm is called probabilistic when there is a probability $\epsilon > 0$ of making an error, e.g. by not accusing guilty users or by accusing innocent users. Later, when we will come to probabilistic schemes, we will elaborate on the exact definition and what the value ϵ really means.

2.2.2 Static versus dynamic

Besides the choice of using a deterministic or a probabilistic scheme, we also distinguish between whether the fingerprinting scheme uses one or multiple sequential fingerprinting codes. In **static schemes**, we assume the distributor generates and distributes one single fingerprinting code, and when at some point a forgery is discovered, some tracing algorithm determines which users should be accused. This is the most practical approach for distributors; this even works when e.g. some single on-demand movie is illegally distributed by a user, as this one forgery is enough to identify this guilty user. A somewhat less practical approach, but giving distributors more freedom, is the class of **dynamic schemes**. After distributing the code and receiving the forgery, distributors are then allowed to start a new **round** of fingerprinting codes, possibly with some suspected pirates disconnected from further participation in the scheme. This allows distributors to gather feedback in the process, and use this information to adjust the next rounds of fingerprinting codes accordingly.

For the dynamic schemes we introduce the concept of **time**, often denoted by the variable t. This variable indicates the round the scheme is currently in. However, using time works only when it is indeed a reasonable assumption that there will be a lot of feedback from pirates. This may work for live streams and pirates broadcasting control words to watch these streams (so that the distributor can detect this illegal distribution real-time and adjust the assignment of control words for the next round) but this does not work when some user distributes, say, an on-demand movie online, several weeks after its release. A dynamic scheme that requires 100 rounds to capture traitors would then require a coalition of pirates to first distribute 100 movies before being able to catch these colluders. And if pirates only distribute these movies after all 100 movies are released, then still a dynamic scheme would not work, as the fingerprints cannot be adjusted in between rounds then. So whether using a dynamic scheme is even an option depends on the scenario, while static schemes can always be used.

Figures 2.1 and 2.2 model the outline of static and dynamic schemes in general. In each of these schemes one can visualize the left part as the distributor, the middle part as the users and the right part as the pirates. The secret data is assumed to be known to the distributor only. This of course includes the distribution of codewords among users, but e.g. in the Tardos scheme this also includes the values of p_i, and in the Boneh-Shaw scheme this includes the secret permutation of columns. The blue blocks in Figure 2.2 represent different rounds. Note that after each round the distributor could of course already choose to disconnect a traitor, if he is convinced of his guilt.

Combining the concepts of deterministic/probabilistic schemes and static/dynamic schemes, we get four main classes of schemes, which are exactly the topics of the following four chapters. In Chapters 4 and 5 we will look at deterministic and probabilistic static schemes respectively, while in Chapters 6 and 7 we will investigate the two classes of dynamic schemes.

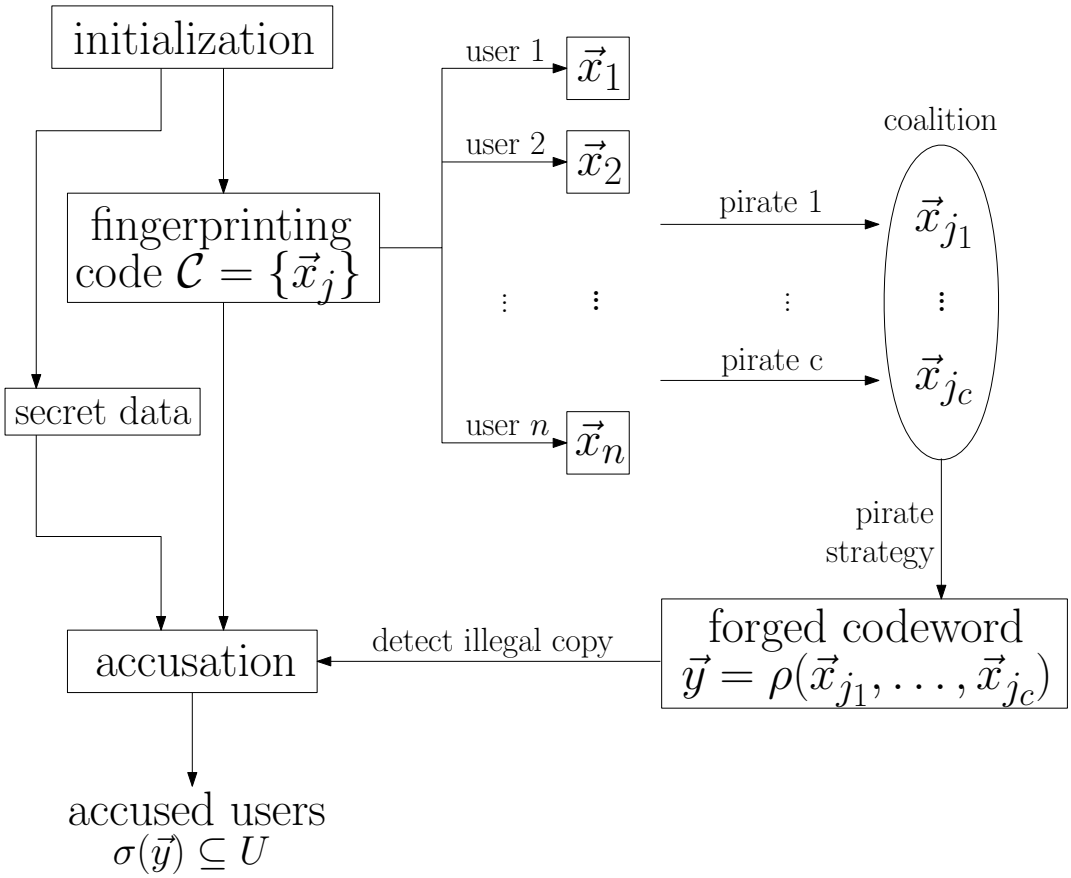

Figure 2.1: *A schematic outline of static schemes. After the initialization, the distributor generates the codewords for each user. These are embedded in the data, and the watermarked data is sent to the users. Some of these users collude to create a forged copy from their own content. This forged copy is distributed online, and the distributor also sees this distributed copy. By running a tracing algorithm σ on this copy, the complete code and perhaps some secret data, the distributor calculates some set $\sigma(\vec{y})$ of guilty users.*

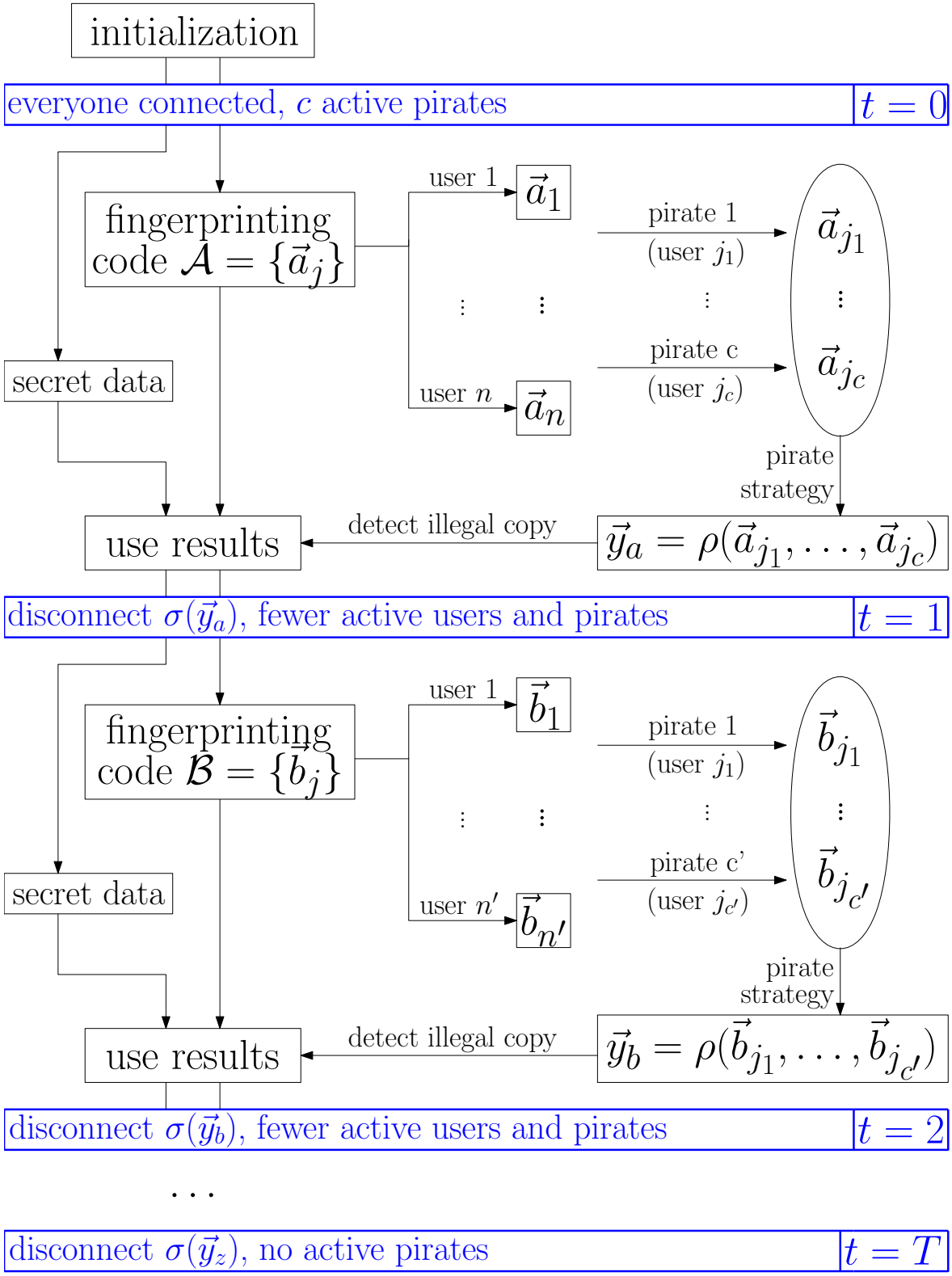

Figure 2.2: *A schematic outline of dynamic schemes. After the initialization phase, the scheme starts with the first round of codewords to send to the users. The pirates get together, create a forgery, and distribute it online. The distributor detects this forgery in real-time, and uses the results from this round to disconnect some users and set up a next fingerprinting code. The disconnected users no longer receive any codewords, so basically the size of the coalition is reduced if some of the pirates have been disconnected. Finally, after several rounds of fingerprinting codes and disconnecting users, the scheme has disconnected all pirates. Thus if one were to start a new round, there would be no forgery $\vec{y}$ anymore.*

2.3 Notation

As we have given a lot of notation above and we will be using most of it extensively, we conclude this chapter with an overview of the most important notation we will use.

$\mathcal{C}$	the fingerprinting code
$\vec{x}_j$	the codeword assigned to user j
$\vec{y}$	a forgery
Q	the alphabet
q	the alphabet size
?	the symbol denoting erasure
$*$	any symbol from the alphabet
n	the total number of users
t	the time
ℓ	the codelength
U	the set of all users, usually taken as $\{1, \ldots, n\}$
C	a coalition of pirates
c	the number of users in a coalition
X	the $\ell \times n$ fingerprinting code matrix
ρ	a pirate strategy
σ	the accusation algorithm

Chapter 3

Preliminaries

Here we will also discuss some preliminaries from several areas of mathematics. This is both for giving a short recap of the building blocks used throughout the report, and for describing which notation we will use in those areas.

3.1 Coding theory

In coding theory we work with vector spaces of the form Q^ℓ, with Q a finite alphabet of some order q and ℓ the length of vectors in this vector space. In these vector spaces we then use the distance function or metric known as the **Hamming distance**. The Hamming distance between two vectors $\vec{x}$ and $\vec{y}$, written as $d(\vec{x}, \vec{y})$, is defined as the number of positions i for which $\vec{x}_i \neq y_i$. It can be shown that this is a metric, and among others it satisfies $d(\vec{x}, \vec{y}) = d(\vec{x} - \vec{y}, \vec{0})$, where $\vec{0}$ is the all-zero vector and the subtraction is done modulo q. This means that the distance between $\vec{x}$ and $\vec{y}$ is equal to the **weight** of $\vec{x} - \vec{y}$, where the weight weight$(\vec{z})$ of some vector $\vec{z}$ is defined as the number of non-zero positions of $\vec{z}$.

Using this metric we define an **error-correcting code** $\mathcal{C}$ of length ℓ, cardinality K and minimum distance d, usually denoted by $(\ell, K, d)_q$ as a subset of K vectors from Q^ℓ such that any two of its **codewords** (vectors $\vec{c}_1, \vec{c}_2 \in \mathcal{C}$) satisfy $d(\vec{c}_1, \vec{c}_2) \geq d$. In other words, if we were to draw spheres with radius less than d around each codeword, then any sphere will contain exactly one codeword. Or similarly, if we draw spheres of radius less than $d/2$ around each codeword, then no two spheres intersect or touch. These error-correcting codes are generally used for correcting errors from data that possibly has some noise (errors) in it. More precisely, if Alice sends a codeword $\vec{c} \in \mathcal{C}$ to Bob, and Bob receives some word $\vec{c}_0$, then Bob can correctly **decode** this word $\vec{c}_0$ to the actual codeword $\vec{c}$ by looking for the nearest codeword, if there were less than $d/2$ errors in $\vec{c}_0$. Or equivalently, Bob can correctly decode the message if $d(\vec{c}_0, \vec{c}) \leq \lfloor (d-1)/2 \rfloor$.

One important class of error-correcting codes is the class of **linear error-correcting codes**, which are codes that form a vector space on their own. This comes down to the extra constraints that (i) any scalar multiple of a codeword is also a codeword (if $\vec{c} \in \mathcal{C}$ and $\alpha \in Q$ then $\alpha\vec{c} \in \mathcal{C}$) and (ii) any sum of two codewords is also a codeword (if $\vec{c}_1, \vec{c}_2 \in \mathcal{C}$ then also $\vec{c}_1 + \vec{c}_2 \in \mathcal{C}$). Since such a code forms a vector space of some dimension k, its cardinality is of the form q^k. The notation used for linear codes is therefore slightly different from non-linear codes: one writes $[\ell, k, d]_q$ to denote a linear error-correcting code of length ℓ, dimension k and minimum distance d. So a linear code with parameters $[\ell, k, d]_q$ has cardinality q^k, while a code with parameters $(\ell, K, d)_q$ has cardinality K.

Since a linear error-correcting code $\mathcal{C}$ forms a vector space of dimension k, such a code is the linear span of k independent base vectors of length ℓ. This means that any codeword from $\mathcal{C}$ is

a linear combination of these k independent vectors. Using a **generator matrix** G, with as rows these k independent vectors, the code $\mathcal{C}$ is equal to the collection of vectors $\vec{c} = \vec{x}G$, with $\vec{x} \in Q^k$. This means we can represent the code by this matrix G rather than having to list all codewords. If k is close to ℓ, then an even shorter representation can be given using the **parity check matrix** H of $\mathcal{C}$. This matrix has $\ell - k$ rows and ℓ columns, where the rows span the complement of the vector space $\mathcal{C}$ in Q^ℓ. Then a word $\vec{c} \in Q^\ell$ is a codeword if and only if it is orthogonal to this complement, which translates to the simple equation $H\vec{c} = \vec{0}$. In this way it is easy to verify whether a word is a codeword, and these matrices can also be used to determine $\vec{c}$ from $\vec{c}_0$ efficiently.

Using basic combinatorics, one can easily prove that any linear code has to satisfy the **Singleton bound** $k + d \leq \ell + 1$, or equivalently $k \leq \ell - d + 1$. Since we want to have as many codewords as possible, given some codelength and minimum distance, a code would be optimal if $k = \ell - d + 1$. Codes satisfying this bound are called **MDS (maximum distance separable) codes**, while codes not satisfying this bound are also called **non-MDS codes**.

From a linear error-correcting code $\mathcal{C}$ we construct a **dual code** $\mathcal{D}$ by saying $\mathcal{D}$ contains exactly those words that are orthogonal to all codewords from $\mathcal{C}$. This means that $\vec{d} \in \mathcal{D}$ if and only if $\vec{d}^T\vec{c} = \vec{0}$, or equivalently D has as parity check matrix G and as generator matrix H. One can then also prove that the dual code of an MDS code with parameters $[\ell, k, d]_q$ is also an MDS code with parameters $[\ell, d-1, k+1]_q$, while the dual of a non-MDS code is also a non-MDS code.

Finally let us conclude this subsection on coding theory with some commonly used codes, which we may also run into here. The Hamming code is perhaps the best known code. Given some parameters $r \geq 2$ and alphabet size q, the q-ary **Hamming code** is the code with parity check matrix consisting of r rows and $\ell = (q^r - 1)/(q - 1)$ columns, such that no two columns are dependent. This code has parameters $[\ell, \ell - r, 3]_q$ and corrects one error. The dual code of this code, which has this parity check matrix as its generator matrix instead, is known as the **simplex code** with parameters $[\ell, r, q^{r-1}]_q$. One particular Hamming code is often referred to as *the* Hamming code, namely the Hamming code with $r = 3$ and $q = 2$, i.e. the code with parameters $[7, 4, 3]_2$. This code is spanned by the rows $\{(1,0,0,0,1,1,0), (0,1,0,0,0,1,1), (0,0,1,0,1,1,1), (0,0,0,1,1,0,1)\}$.

Another code we will run into in this report is the following. Given a field Q of prime order q, a primitive element $\alpha \in Q$ and numbers k and ℓ, we construct the **Reed-Solomon code** of length ℓ and dimension k by $\mathcal{C} = \{(f(0), f(1), f(\alpha), f(\alpha^2), \ldots, f(\alpha^{\ell-2})) \mid f \in \mathbb{F}[X], \deg(f) < k\}$, where $\mathbb{F}[X]$ is the set of polynomials in X with coefficients from $\mathbb{F}$. This code is a linear code with parameters $[\ell, k, \ell - k + 1]_q$ and is thus an MDS code.

3.2 Probability theory

Besides coding theory, we will also run into basic probability theory quite often. Of course this happens mostly in the chapters on probabilistic schemes, where we somehow have to prove that the probability of making an error is sufficiently small. First of all, we write the **probability** that event E occurs as $\mathbb{P}[E]$, e.g. $\mathbb{P}[X = 3] = 1/6$ if X is the outcome of a fair dice roll. Also using this math blackboard style, we write $\mathbb{E}[X]$ for the **expectation** of X, i.e. $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}[X = x]$ if X is a discrete random variable on $\mathcal{X}$, and $\mathbb{E}[X] = \int_{\mathcal{X}} xf(x)dx$, if X is a continuous random variable on $\mathcal{X}$ with **probability density function** (pdf) $f(x)$ of X. We will also use standard rules for the expectation, such as $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$ and $\mathbb{E}[aX] = a\mathbb{E}[X]$ for random variables X, Y and scalars a. We also use the **cumulative distribution function** (cdf) of a continuous random variable, usually denoted by $F(x)$, which is defined by $F(x) = \int_{-\infty}^{x} f(x)dx$. In particular $F(-\infty) = 0, F(+\infty) = 1$ and $f(x) = \frac{d}{dx}F(x)$ if f is continuous at x. Finally we will also use the **variance** of a random variable, defined by

$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$. And when using statistical data, we often denote the statistical mean and variance of a data set by μ and σ^2 respectively.

One **probability distribution** we will run into in this report is the **Bernoulli distribution**, which is arguably the most simple of all non-trivial distributions. Given some probability of success p, and a complementary probability of failure $1 - p$ and denoting success by 1 and failure by 0, a random variable X with a Bernoulli distribution with parameter p satisfies $\mathbb{P}[X = 1] = p$ and $\mathbb{P}[X = 0] = 1 - p$. We will denote this distribution by $\text{Ber}(p)$. Another simple distribution is the distribution of the number of successes in n independent and identical Bernoulli trials, which is known as the **binomial distribution**. This distribution function has the form $\mathbb{P}[X = x] = \binom{n}{x} p^x (1-p)^{n-x}$ for any $0 \leq x \leq n$. We sometimes abbreviate the binomial distribution of n trials and success probability p by $\text{Bin}(n, p)$. Related to this distribution is the **negative binomial distribution**, which counts the number of successes until a fixed number r of failures arise, where each trial gives a success with probability p. This distribution has mean $rp/(1-p)$ and variance $rp/(1-p)^2$.

Finally the only often considered distribution used in this report is the **normal distribution**, also known as the **Gaussian distribution**. Given some distribution function, one can generate samples according to this distribution, and calculate the mean for each of these samples. If we then were to plot a histogram of these sample means, one would get a peak around the real mean of the distribution, and the shape will be somewhat bell-curved. Letting the number of trials and the individual sample sizes go to infinity, this histogram will always take the same shape. This is known as the **central limit theorem**, and the resulting distribution function is a (scaled, translated) Normal distribution. A normal distribution with mean μ and variance σ^2 has as pdf the function $f(x) = (2\pi\sigma^2)^{-1/2} e^{-(x-\mu)^2/2\sigma^2}$. We denote the normal distribution with mean μ and variance σ^2 by $\mathcal{N}(\mu, \sigma^2)$.

We conclude this subsection with two somewhat more advanced results, which are useful for bounding probabilities. The **Markov inequality** states that given some non-negative random variable X and some value $a \neq 0$, it always holds that $\mathbb{P}[X \geq a] \leq \mathbb{E}[X]/a$. This can easily be proven using the **indicator function** defined as $\mathbb{I}[E] = 1$ if E occurs and $\mathbb{I}[E] = 0$ otherwise. Then $a \cdot \mathbb{I}[X \geq a]$ is 0 whenever $X < a$ and $a \cdot \mathbb{I}[X \geq a]$ is a if $X \geq a$, hence $a \cdot \mathbb{I}[X \geq a] \leq X$. So $\mathbb{E}[X] \geq \mathbb{E}[a \cdot \mathbb{I}[X \geq a]] = a \cdot \mathbb{E}[\mathbb{I}[X \geq a]] = a \cdot (1 \cdot \mathbb{P}[X \geq a] + 0 \cdot \mathbb{P}[X \leq a]) = a \cdot \mathbb{P}[X \geq a]$, which proves the result.

The other result we use is the **Chernoff bound**, which in fact makes use of the Markov inequality. One particular instance of the bound which we will use says that given a binomial random variable X on n trials, with probability of success $1/2$ and mean $\mu = \mathbb{E}[X] = n/2$, the probability of large deviation of X from its mean is bounded by $\mathbb{P}[X - \mu \geq a] \leq e^{-2a^2/n}$ for any $a > 0$. So the probability of a large deviation of such a binomial random variable from its mean decreases exponentially, with a quadratic term a^2 in the exponent.

3.3 Graph theory

Later on in the report we will also make use of some basic graph theory, so we will also discuss those basics here. First of all, an **(undirected) graph** G is defined by a set of points V, also known as the **vertices**, and a set of lines E connecting pairs of vertices, also known as **edges**. We also write $G = (V, E)$ to describe this graph. To describe that an edge runs from some vertex u to another vertex v there are several notations, e.g. $u \sim v, u \leftrightarrow v$, or simply by an unordered set $\{u, v\}$. In this report we will also consider **graph colorings**, which are functions from V to some set of colors K, such that each vertex gets exactly one color. These colors are also often simply represented by numbers $1, \ldots, k$.

There are several properties a graph can have which are all very interesting, but we will only use a few of these terms. We refer to the **degree** of a vertex v as the number of edges going through v, i.e. the number of **neighbors** v has. We will also denote the degree of v by $d(v)$. Furthermore a **matching** is a subset of the edges $M \subseteq E$ such that in the subgraph $G' = (V, M)$, each vertex has degree at most 1. A **maximum matching** is a matching of maximum cardinality. Strongly related to matchings are **vertex covers**, which are subsets $W \subseteq V$ such that all edges in E have at least one of the vertices of W as one of its endpoints.

3.4 Miscellaneous

Finally we list some miscellaneous terminology which is used throughout this report. We use the **big Oh notation** to describe growth rates of functions of algorithms, as its parameters tend to infinity. For example, we may write $f(x) = \mathcal{O}(g(x))$ to say that as x goes to infinity, $f(x)$ grows as most as fast as some constant times $g(x)$. More precisely, there exist some positive constants M and x_0 such that $|f(x)| \leq M|g(x)|$ for all $x \geq x_0$. We also use the **big Omega notation**, $f(x) = \Omega(g(x))$, to denote that $|f(x)| \geq M|g(x)|$ as $x \to \infty$ for some constant M. Combining both gives the **big Theta notation** $f(x) = \Theta(g(x))$, which means that there exist two constants M_1, M_2 such that $M_1|g(x)| \leq |f(x)| \leq M_2|g(x)|$ as $x \to \infty$. In some cases it is also convenient to use the **small oh notation**, which says that $f(x) = o(g(x))$ if $\lim_{x\to\infty} f(x)/g(x) = 0$. This is a slightly stronger statement than $f(x) = O(g(x))$, e.g. $x^2 = \mathcal{O}(x^a)$ for any $a \geq 2$, while $x^2 = o(x^a)$ only for $a > 2$.

Lastly we borrow from information theory the basic terms entropy and mutual information. The **entropy** of a discrete random variable X, written as $H(X)$, is defined as $H(X) = -\sum_{x\in\mathcal{X}} x \log_2(\mathbb{P}[X = x])$. The entropy basically describes the amount of information gained from taking a sample from X; if one can easily guess the outcome with high success rate, e.g. $X \sim \text{Ber}(1-\epsilon)$, then $H(X) \approx 0$, while if $X \sim \text{Ber}(1/2)$ we get $H(X) = 1$. Also from information theory we use the term **mutual information**. Given two random variables X, Y, the mutual information between these two variables, written as $I(X, Y)$ or $I(X; Y)$, is defined as $I(X; Y) = \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \mathbb{P}[X = x, Y = y] \log_2(\mathbb{P}[X = x, Y = y]/\mathbb{P}[X = x]\mathbb{P}[Y = y])$. This value measures the mutual dependence between the two random variables; it tells us how much knowing one random variable will tell us about the other. If X and Y are totally dependent, then knowing X gives us a lot of information about Y and thus $I(X; Y)$ is large. If however X and Y are independent, then the term inside the logarithm is 1, so that the logarithm is always 0 and $I(X; Y) = 0$. So then knowing X tells us nothing about Y, as expected.

Part I

Literature survey

Chapter 4

Deterministic static schemes

***Citations**: For writing this chapter, the following articles were used: [AFS01], [AS04], [BCE+01], [Bla03a], [Bla03b], [Bla03c], [BEN07], [BS98], [CE00], [EC01], [EC02], [HVLLT98], [Ker10], [SS01], [SSW01], [SW98], [STW00], [SZ08], [TM05], [XMS07]*

4.1 Introduction

In this chapter we will discuss deterministic static schemes, which are schemes using a single fingerprinting code (static) and a deterministic tracing process. The latter means that a scheme will only accuse users if there is certainty about guilt. This definition is still unclear and ambiguous, and so there is still room for different interpretations of what properties such a code should have, resulting in different classes of fingerprinting codes. We will discuss some of the classes of codes which are most widely studied in literature in this chapter. Note that throughout this chapter we will always be working in the restricted digit model, so that the pirates can produce $\langle \vec{x}_1, \ldots, \vec{x}_k \rangle := \{\vec{y} \in Q^\ell \mid \forall 1 \leq i \leq n : (\vec{y})_i \in \{(\vec{x}_1)_i, \ldots, (\vec{x}_k)_i\}\}$.

First of all, there is the easiest and most basic definition of which properties a code should at least have, namely resistance against framing attacks. This means that a coalition should not be able to produce a codeword belonging to a user outside the coalition, since then the whole fingerprinting code would be useless; then even if a single pirate outputs his copy, we cannot accuse him with certainty, since he could argue that he was framed. Codes with the no-framing property are called **frameproof codes**, and they were studied in [Bla03c], [BS98], [CE00], [EC02], [SW98].[1]

Definition 4.1 (Frameproof codes)**.** *Let $\mathcal{C} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ be a fingerprinting code, and let $c \geq 1$. Then $\mathcal{C}$ is called a **c-frameproof code** if no coalition of size at most c can generate a codeword $\vec{x} \in \mathcal{C}$ belonging to a user outside the coalition.*

Another characterization of frameproof codes is that for any codeword not in some coalition of size at most c, there exists some position i such that the i^{th} symbol of this codeword is different from all symbols seen by the coalition on that position; if for some codeword there exists no such position, then the code is not frameproof, while if there is always such a position the code is obviously frameproof.

Example 4.2 (A binary 2-frameproof code)**.** *Let $\ell = 3$, let $q = 2$, and let $n = 4$. Suppose $\mathcal{C} = \{000, 011, 101, 110\}$ as in Table 4.1. In this code, any two codewords have Hamming*

[1] An even weaker class of codes was studied in [AB08], namely **randomized frameproof codes**. These are codes which are frameproof with high probability, and with a small probability these codes may not even be frameproof. In this chapter we do not study these codes, but for completeness we mention the existence of such codes here.

distance 2*, which implies that* $\langle \vec{x}_1, \vec{x}_2 \rangle \neq Q^3$ *for any two codewords* $\vec{x}_1, \vec{x}_2 \in \mathcal{C}$*. Also, any three codewords can generate all words of length* 3*, i.e.* $\langle \vec{x}_1, \vec{x}_2, \vec{x}_3 \rangle = Q^3$ *for any three codewords* $\vec{x}_1, \vec{x}_2, \vec{x}_3 \in \mathcal{C}$*. Thus if some codeword* $\vec{x}_3 \in \mathcal{C}$ *could have been generated by some coalition possessing only two other codewords* $\vec{x}_1, \vec{x}_2$ *both different from* $\vec{x}_3$*, i.e. if* $\vec{x}_3 \in \langle \vec{x}_1, \vec{x}_2 \rangle$*, then we would have* $Q^3 \neq \langle \vec{x}_1, \vec{x}_2 \rangle = \langle \vec{x}_1, \vec{x}_2, \vec{x}_3 \rangle = Q^3$ *which is obviously a contradiction. So this code is* 2*-frameproof.*

Alice	0	0	0
Bob	0	1	1
Charlie	1	0	1
Dave	1	1	0

Table 4.1: *The fingerprinting code* $\mathcal{C}$ *for* 4 *users with length* $\ell = 3$*.*

Note that in the above example, there is no **traceability**. Given the forgery, one cannot always find a member of the coalition. In fact, using e.g. the majority voting strategy pirates will always get away with piracy in this example. For instance the word 001 could have been generated by the three pairs of codewords $\{000, 101\}, \{000, 011\}, \{101, 011\}$ while $\{000, 101\} \cap \{000, 011\} \cap \{101, 011\} = \emptyset$. So no user is part of all suspicious coalitions. Therefore any accusation of a single user may be a false accusation, which makes it impossible to accuse any guilty user with certainty.

Since frameproof codes do not have any requirements about traceability besides the case that $\vec{y} \in \mathcal{C}$, which makes these codes impractical for traitor tracing purposes, in the literature one therefore also considers a somewhat stronger class of frameproof codes, namely the class of **secure frameproof codes**. These were studied in e.g. [EC02], [STW00], [SZ08].

Definition 4.3 (Secure frameproof codes)**.** *Let* $\mathcal{C} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ *be a fingerprinting code, and let* $c_1, c_2 \geq 1$*. Then* $\mathcal{C}$ *is a* (c_1, c_2)***-secure frameproof code*** *if and only if no two disjoint coalitions* C_1, C_2 *of size at most* c_1, c_2 *respectively, can generate a common word* $\vec{x}$*. Sometimes* $c_1 = c_2 = c$*, in which case we simply say that* $\mathcal{C}$ *is a* ***c-secure frameproof code****.*

Example 4.4 (A binary 2-secure frameproof code)**.** *Again consider the code* $\mathcal{C} = \{000, 011, 101, 110\}$ *from Example 4.2 and Table 4.1, with parameters* $\ell = 3, q = 2, n = 4$*. This code is a* 2*-secure frameproof code, since no two disjoint coalitions of size at most* 2 *can generate a common word. To see this, first notice that if one of the disjoint coalitions has size only* 1*, then there is no common word because, as we saw earlier, the code is* 2*-frameproof. For two coalitions of size* 2*, one can simply check all possibilities. Consider for example the two disjoint coalitions* C_1, C_2 *possessing the codewords* $\{000, 101\}$ *and* $\{011, 110\}$ *respectively. Then* $\langle 000, 101 \rangle = \{000, 001, 100, 101\}$ *and* $\langle 011, 110 \rangle = \{010, 011, 110, 111\}$*, hence* $\langle 000, 101 \rangle \cap \langle 011, 110 \rangle = \emptyset$ *as required.*

Note that frameproof codes are a special case of secure frameproof codes; a code is c-frameproof if and only if the code is $(c, 1)$-secure frameproof. Also, note that there is a simple ordering in strength of secure frameproof codes. If a code is (c_1, c_2)-secure frameproof, then it is also trivially (c_1', c_2')-secure frameproof for any $c_1' \leq c_1, c_2' \leq c_2$. In particular, if a code is c-secure frameproof, then it is also c-frameproof.

Strongly related to secure frameproof codes are so-called **separating hash families**, which are defined as follows.

Definition 4.5 (Hash families and separating hash families)**.** *Let* U, Q *be sets of cardinality* n, q *respectively. Then a collection* $\mathcal{F}$ *of* ℓ *functions from* U *to* Q *is an* (ℓ, n, q)***-hash family****.*

Furthermore for natural numbers c_1, c_2 an (ℓ, n, q)-hash family is called an $(\ell, n, q, \{c_1, c_2\})$- ***separating hash family*** *if for any two disjoint subsets C_1, C_2 of U of cardinality at most c_1 and c_2 respectively, there exists at least one function $g \in \mathcal{F}$ such that $\{g(a) : a \in C_1\} \cap \{g(a) : a \in C_2\} = \emptyset$.*

For certain parameters, separating hash families and secure frameproof codes are equivalent. More precisely, if we identify elements of U with users, functions $g_i \in \mathcal{F}$ with fingerprinting positions i, vectors $\vec{g}_j = (g_1(a_j), \ldots, g_\ell(a_j)) \in Q^\ell$ with codewords assigned to users j, and elements of Q with symbols of an alphabet of size q, then the resulting code is a c-secure frameproof code, as the following result states.

Theorem 4.6 (Relation between separating hash families and secure frameproof codes)**.** *Let $\mathcal{F}$ be an (ℓ, n, q)-hash family with $n \geq 2c$. Then the fingerprinting code associated to $\mathcal{F}$ is a c-secure frameproof code if and only if the hash family $\mathcal{F}$ is an $(\ell, n, q, \{c, c\})$-separating hash family.*

Some papers investigated the problem of finding large $(\ell, n, q, \{c, c\})$-separating hash families instead of finding large secure frameproof codes. Here we will not consider these separating hash families separately, since they are not of interest to us. We will only mention the results obtained via these separating hash families about secure frameproof codes.

Secure frameproof codes still do not permit traceability, but there is added security. For example, a coalition C of size at most c cannot implicate a disjoint coalition C' of size at most c by generating a word that could also have been generated by C'. Then two disjoint coalitions would have a nonempty intersection of their feasible sets, contradicting the fact that the code is c-secure frameproof. This implies that with a c-secure frameproof code, if a forgery $\vec{y}$ is obtained and it is known that some coalition C' of size at most c could have generated $\vec{y}$, then C' contains at least one traitor. This gives a trivial tracing algorithm with success rate $1/c$, which first calculates which coalitions of size at most c could possibly have generated $\vec{y}$, randomly selects one of these and then randomly selects one of its users. In some cases a better success rate than $1/c$ can be achieved. For example, in the case $c = 2$ the following result was proven in [STW00].

Theorem 4.7 (Traceability of 2-secure codes)**.** *Let $\mathcal{C}$ be a 2-secure frameproof code, and let $\vec{y}$ be a forgery generated by some coalition C. Then either a guilty user can be identified, or a set of three users can be identified, two of which must be guilty.*

Thus, depending on which case arises, either a success rate of accusations of 1 or 2/3 can be achieved for the case $c = 2$. As we will see later, for large c we conjecture that the worst-case rate is approximately $1/c$ for large c, which makes these codes impractical for the use with a tracing algorithm. Therefore we need an even stronger class of fingerprinting codes, which really allows for tracing traitors. In the literature these codes are known as **IPP codes**, which were first introduced in [HVLLT98] and later also studied in [AFS01], [AS04], [BCE+01], [BEN07], [Bla03a], [SS01], [TM05], [XMS07]. [2]

Definition 4.8 (IPP codes)**.** *Let $\mathcal{C} = \{\vec{x}_1, \ldots, \vec{x}_n\} \subseteq Q^\ell$, with $|Q| = q$, and let $c \geq 2$. Then $\mathcal{C}$ has the* ***c-identifiable parent property****, or $\mathcal{C}$ is a* ***c-IPP code****, if there exists a tracing algorithm σ such that if a coalition C of size at most c generates a forgery $\vec{y}$, then σ returns only guilty users (i.e. $\sigma(\vec{y}) \subseteq C$) and σ returns at least one guilty user (i.e. $|\sigma(\vec{y})| \geq 1$).*

[2] The 'IPP' stands for the **identifiable parent property**, which finds its origin in DNA strings. Then codewords translate to DNA-strings of parents, and feasible forgeries translate to possible DNA-strings of children of these parents. The identifiable parent property then means that given the DNA-string of a child and knowing all DNA-strings of possible parents, one can idenfity at least one of the parents. Later the term IPP code was used in the more general sense for c parents, with c not necessarily equal to 2.

Example 4.9 (A ternary 2-IPP code)**.** *Consider the following code, which is also given in Table 4.2:*

$$\mathcal{C} = \{0000, 0111, 0222, 1012, 1120, 1201, 2021, 2102, 2210\}. \tag{4.1}$$

This code has parameters $\ell = 4, q = 3, n = 9$, and is also known as the Tetracode or the ternary Hamming code of length 4. This is a linear error-correcting code with parameters $[4, 2, 3]_3$, so any two codewords have Hamming distance at least 3. One can easily verify that any two codewords indeed have exactly Hamming distance 3, e.g. by verifying that any non-zero codeword has exactly one zero.

Now suppose a coalition of size 2 generates a forgery $\vec{y}$. Then, since the coalition's two codewords match on exactly one position, the codeword $\vec{y}$ matches with both codewords on that one position as well. Also, $\vec{y}$ matches exactly one of them on the remaining three positions, giving 5 matches with the coalition's codewords in total. So one of the two codewords must match the forgery on at least 3 positions, giving it a Hamming distance of at most 1 to the forgery. So the tracing algorithm σ, which takes as input a forgery and outputs the user whose codeword is nearest to the forgery in terms of Hamming distance, will always accuse exactly this guilty user. Therefore this code is a 2-IPP code.

Alice	0	0	0	0
Bob	0	1	1	1
Charlie	0	2	2	2
Dave	1	0	1	2
Eve	1	1	2	0
Fred	1	2	0	1
George	2	0	2	1
Henry	2	1	0	2
Isaac	2	2	1	0

Table 4.2: *The 2-IPP code $\mathcal{C}$ from Example 4.9, also known as the Tetracode. It was shown in [BEN07] that this is one of only few 'beautiful' IPP codes.*

In the hierarchy of deterministic static fingerprinting schemes, these codes are better than all other types of codes mentioned before; any c-IPP code is also a c-secure frameproof code and hence also a c-frameproof code. And with the addition of traceability, these codes are now more suited for the purpose of tracing traitors. However, the price we pay for this traceability, besides possibly longer codelengths and a smaller cardinality of the codes is big, as the following Theorem shows.

Theorem 4.10 (Minimum alphabet size of IPP codes)**.** *Let $\mathcal{C}$ be a c-IPP code with alphabet size q and cardinality $n > c$. Then $q > c$.*

While a longer codelength may be an acceptable consequence of better security, a bigger alphabet size is very much unwanted. In the setting of broadcast channels, for instance, an implementation of one of q symbols of a codeword position may be based on giving the user one of q keys for decrypting a segment of the data. Then the distributor broadcasts the same segment of the content q times, encrypted with the q different keys, so that users with certain keys (symbols) can only decrypt one of those q encrypted segments. Thus an alphabet of size q would require the distributor to distribute the same segment of the content q times. On the other hand, the length of the codeword translates to the number of segments the content is divided in, which does not really increase the bandwidth needed. Thus one would rather have a long codelength than a big alphabet.

Some papers investigated an even stronger class of codes, called **traceability codes**. These are defined as follows.

Definition 4.11 (Traceability codes)**.** *Let $\mathcal{C} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ be a fingerprinting code, and let $c \geq 2$. Then $\mathcal{C}$ is a* ***c-traceability code*** *if for any coalition of size at most c and any forgery generated by such a coalition, the user whose codeword has the smallest Hamming distance to the forgery is always a guilty user.*

These traceability codes have the added advantage that the tracing algorithm is simple: one just calculates which of the codewords has the smallest Hamming distance to the forgery, and accuses the user associated to that codeword. However, the complexity of the tracing algorithm is not the biggest problem of IPP codes and deterministic static schemes in general. These traceability codes thus solve a problem that is not really a big problem, while maintaining or even increasing the problem of a large codelength and a big alphabet size. Therefore we will not consider these codes here. Note however that in particular with traceability codes, the problem clearly asks for the use of error-correcting codes, as with error-correcting codes one also 'decodes' errors by looking for the nearest neighbor in the ℓ-dimensional space using the Hamming distance as the metric. We will see that for the other classes of codes mentioned above, error-correcting codes also show up more than once.

The rest of this chapter is structured as follows. First, in Section 4.2 we give an overview of the most important results about frameproof codes that can be found in literature. Then, in Section 4.3 we do the same for secure frameproof codes. In Section 4.4 we then investigate IPP codes, and mention bounds and constructions known for this class of codes. Finally in Section 4.5 we summarize the results from this chapter.

4.2 Frameproof codes

As we saw above, c-frameproof codes are codes with only one requirement: no c users can frame a single innocent user by generating his codeword from their codewords. Still, because of this one requirement it is already quite difficult to construct large codes with short codelengths and small alphabet sizes. Solving this problem of getting a maximum code size for a given codelength and alphabet size, or minimizing the codelength for a given cardinality and alphabet size is already a non-trivial problem, and this problem has only been solved for a few special cases. For notation convenience, we will write $F(c, \ell, q)$ throughout this section to denote the maximum cardinality of a c-frameproof code of length ℓ and alphabet size q.

Let us start with an important bound derived in [Bla03c], which, as we will see later, is (almost) tight for certain values of c, ℓ, q.

Theorem 4.12. *[Bla03c, Theorem 1 and Corollary 2] Let $c, \ell, q \geq 2$, let $r \in \{0, \ldots, c-1\}$ such that $r \equiv \ell \mod c$ and let $k_1 = \lceil \ell/c \rceil$ and $k_0 = \lfloor \ell/c \rfloor$. Then*

$$F(c, \ell, q) \leq \max\{q^{k_1}, r(q^{k_1} - 1) + (c - r)(q^{k_0} - 1)\} \leq \max\{1, r\}q^{k_1} + \mathcal{O}(q^{k_1 - 1}). \tag{4.2}$$

From this it follows that $\ell = \Omega(c \log_q(n))$.

Proof. Let C be a frameproof code with parameters (c, ℓ, q). For any subset of positions $S \subseteq \{1, \ldots, \ell\}$, define $U(S)$ as the set of all codewords $\vec{x} \in C$ which are 'unique' on the positions identified by S, i.e. let $U(S) = \{\vec{x} \in C \mid \forall \vec{y} \in C \setminus \{\vec{x}\}, \vec{y}[S] \neq \vec{x}[S]\}$, where $\vec{x}[S] = (x_i)_{i \in S}$, e.g. the subword of $\vec{x}$ indexed by positions in S. Then obviously $|U(S)| \leq q^{|S|}$, since every codeword $\vec{x} \in U(S)$ is uniquely identified by its subword $\vec{x}[S]$, for which there are only $q^{|C|}$ different choices.

Also, if this bound is met, then every subword $\vec{s} \in Q^{|S|}$ corresponds to one unique codeword $\vec{x} \in C$ with $\vec{x}[S] = \vec{s}$. Hence, if $|C| > q^{|S|}$, then at least one vector $\vec{s} \in Q^{|S|}$ corresponds to at least two vectors $\vec{x}, \vec{y} \in C$ with $\vec{x}[S] = \vec{y}[S] = \vec{s}$, so that $|U(S)| \leq q^{|S|} - 1$.

Let $S_1, \ldots, S_c \subseteq \{1, \ldots, \ell\}$ be a partition of $\{1, \ldots, \ell\}$, with each subset containing approximately ℓ/c elements (i.e. r of them have size $\lceil \ell/c \rceil = k_1$ and $c - r$ of them have size $\lfloor \ell/c \rfloor = k_0$). If $|C| > q^{k_1}$ then $|C| > q^{|S_j|}$ for all of these sets S_j, so that $|U(S_j)| \leq q^{|S_j|} - 1$. So then $\sum_{j=1}^{c} |U(S_j)| = r(q^{k_1} - 1) + (c - r)(q^{k_0} - 1)$ gives us exactly the bound we need. Thus proving that $C = \cup_{j=1}^{c} U(S_j)$ proves that $|C| = |\cup_{j=1}^{c} U(S_j)| \leq \cup_{j=1}^{c} |U(S_j)| = r(q^{k_1} - 1) + (c - r)(q^{k_0} - 1)$, which would prove the result.

Now assume that $C \neq \cup_{j=1}^{c} U(S_j)$, e.g. $\vec{x} \in C \setminus (\cup_{j=1}^{c} U(S_j))$. Since $\vec{x}$ is not in any of these sets $U(S_j)$, $\vec{x}$ is not unique on these sets of positions, so there exist $\vec{x}_1, \ldots, \vec{x}_c \in C$ with $\vec{x}[S_j] = \vec{x}_j[S_j]$ for all $j \in \{1, \ldots, c\}$. But then $\vec{x}$ can be generated by the coalition of size c possessing the codewords $\{\vec{x}_1, \ldots, \vec{x}_c\}$, which is a contradiction with the assumption that C is frameproof against c colluders. This proves the result. □

This bound allows us to solve $F(c, \ell, q)$ for a special class of codes, namely codes with $\ell \leq c$. The following code belongs to this special class, as the Lemma after the definition shows.

Definition 4.13. *[BS98, Section IIIA] We define the* ***q-ary unit code*** *of size $\ell(q-1)$ as $\Gamma_q(\ell) = \{\alpha \vec{e}_i \mid \alpha \in Q \setminus \{0\}, 1 \leq i \leq \ell\}$. This code consists of all vectors with $\ell - 1$ zeroes and a single non-zero symbol.*

Lemma 4.14. *[Bla03c, Construction 1] [BS98, Claim III.1, for $q = 2$] Let $c, \ell, q \geq 2$ with $\ell = c$. Then the code $\mathcal{C} = \Gamma_q(\ell)$ is c-frameproof and has cardinality $n = \ell(q-1)$.*

Proof. Let C be given as above, and suppose that C is not frameproof, so that there exists some codeword $\vec{x} = \alpha_0 \vec{e}_{i_0} \in C$ that could have been generated by some other coalition not possessing the codeword $\vec{x}$. Since $\vec{x}$ is the only word in C which has symbol α_0 on position i_0 this immediately leads to a contradiction. Therefore $\Gamma_q(\ell)$ is frameproof for any ℓ. □

As it turns out, there is no better code than this in this special class of codes, which is an immediate consequence of the upper bound on the cardinality of any frameproof code given in Theorem 4.12.

Corollary 4.15. *[Bla03c, Corollary 3] Let $\ell \leq c$. Then $F(c, \ell, q) = \ell(q-1)$.*

An immediate consequence of the previous results is that any binary c-frameproof code of cardinality $n \geq c$ must have length at least c, as was also noted in [BS98, Section IIIA].

The bound given in Theorem 4.12 is not only tight for the above-mentioned special class of codes; for the general case one can also construct codes of cardinality $\Theta(q^{k_1})$, as was proven in [Bla03c]. However, these codes satisfy $q \geq \ell = \Omega(c \log(n))$, thus giving a huge alphabet size.

Theorem 4.16. *[Bla03c, Construction 2] Let $c, \ell, q \geq 2$, and let $q \geq \ell$ be a prime power. Let $k_1 = \lceil \ell/c \rceil$, and let $\alpha_1, \ldots, \alpha_\ell$ be ℓ distinct elements from the finite field $\mathbb{F}_q$. Finally let C be defined as the following (Reed-Solomon) code:*

$$C = \{(f(\alpha_1), \ldots, f(\alpha_\ell)) \mid f \in \mathbb{F}_q[X], \deg(f) < k_1\}. \tag{4.3}$$

Then the code C has cardinality q^{k_1} and is frameproof against c colluders.

Proof. First note that any polynomial f over a finite field $\mathbb{F}_q$ of degree at most $k_1 - 1$ is uniquely determined by specifying f at any k_1 points $\alpha_1, \ldots, \alpha_{k_1}$. Since $k_1 < \ell$, it follows that $|C| = q^{k_1}$. Now suppose C is not frameproof against c colluders, so that some coalition of size c, possessing some codewords $Y = \{\vec{y}_1, \ldots, \vec{y}_c\}$, can generate a codeword $\vec{x} \in C \setminus Y$. Since $\vec{x}$ agrees with elements in Y in a total of ℓ positions, there must be some codeword $y_i \in Y$ which matches $\vec{x}$ in at least $\lceil \ell/c \rceil = k_1$ positions, say positions $S = \{i_1, \ldots, i_{k_1}\}$. But if $\vec{x}[S] = y_i[S]$, then the polynomial f associated to $\vec{x}$ also intersects with the polynomial g associated to $\vec{y}$ on k_1 positions, which implies that $f = g$, hence $\vec{x} = y_i$. This is a contradiction with $\vec{x} \in C \setminus Y$, hence the result. □

	$f(\alpha_1)$ $(\alpha_1 = 0)$	$f(\alpha_2)$ $(\alpha_2 = 1)$	$f(\alpha_3)$ $(\alpha_3 = 2)$	
Alice	0	0	0	$(f(X) = 0)$
Bob	0	0	1	$(f(X) = 1)$
Charlie	0	0	2	$(f(X) = 2)$
Dave	0	1	2	$(f(X) = X)$
Eve	1	2	0	$(f(X) = X + 1)$
Fred	2	0	1	$(f(X) = X + 2)$
George	0	2	1	$(f(X) = 2X)$
Henry	1	0	2	$(f(X) = 2X + 1)$
Isaac	2	1	0	$(f(X) = 2X + 2)$

Table 4.3: *The 2-frameproof code constructed from Theorem 4.16 by taking $q = \ell = 3$ and $c = 2$, so that $k_1 = 2$. The polynomials $f(X)$ are of degree at most 1, so we have 9 polynomials $f(X) = aX + b$ for $a, b \in \mathbb{F}_3$. The values of α are simply taken as $\alpha_1 = 0, \alpha_2 = 1, \alpha_3 = 2$.*

For even bigger alphabet sizes, it was shown that the other value of the maximum in Theorem 4.12 is also (almost) tight. For the case that $k_1 = k_0$, i.e. ℓ/c integral, and $\ell \equiv 0 \mod c$, Theorem 4.12 gives an upper bound on the cardinality of a c-frameproof code of $c(q^{\ell/c} - 1)$. The following construction matches the highest order term of this bound for $c = 2$.

Theorem 4.17. *[Bla03c, Construction 3] Let $c = 2$, let $\ell = 2\ell_0 \geq 4$ be even, let $q_0 \geq \ell + 1$ be a prime power and let $q = q_0^2 + 1$. Let $\mathbb{F}_q$ be the finite field of order q_0, and let $F = \{\infty\} \cup (\mathbb{F}_{q_0})^2 = \{\infty, (0,0), (0,1), \ldots, (q_0 - 1, q_0 - 1)\}$ so that $|F| = q$. Let $\beta_0, \beta_1, \alpha_1, \ldots, \alpha_{\ell-1}$ be $\ell + 1$ distinct elements from $\mathbb{F}_{q_0}$. For two polynomials $f, g \in \mathbb{F}_{q_0}[X]$, write $(f, g)(\alpha) = (f(\alpha), g(\alpha))$. Let the codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq F^\ell$ be defined as*

$$\mathcal{C}_1 = \{(\infty, (f,g)(\alpha_1), (f,g)(\alpha_2), \ldots, (f,g)(\alpha_{\ell-1})) \mid \deg(g) \leq \deg(f) = \ell_0 - 1\}, \tag{4.4}$$

$$\mathcal{C}_2 = \{((t(\beta_0), t(\beta_1)), (s,t)(\alpha_1), \ldots, (s,t)(\alpha_{\ell-1})) \mid \deg(s) \leq \ell_0 - 2, \deg(t) \leq \ell_0\}. \tag{4.5}$$

Then the code $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ is 2-frameproof with cardinality $q_0^{\ell-1}(2q_0 - 1) = cq^{\ell/c} - o(q^{\ell/c})$.

Proof. First, remark that $(t(\beta_0), t(\beta_1)) \neq \infty$, so that C_1 and C_2 are disjoint. Hence $|C| = |C_1 \cup C_2| = |C_1| + |C_2|$. Now for C_1 we see that $\vec{x} \in C_1$ is uniquely determined when f and g are specified. Since there are $q_0^{\ell_0 - 1}(q_0 - 1)$ choices for f (q_0 choices for the first $\ell_0 - 1$ coefficients and $q_0 - 1$ non-zero choices for the coefficient of $X^{\ell_0 - 1}$) and $q_0^{\ell_0}$ choices for g, we have $|C_1| = q_0^{2\ell_0 - 1}(q_0 - 1) = q_0^{2\ell_0}(1 - 1/q_0)$. Similarly, determining codewords in C_2 equals determining s and t, for which there are $q_0^{\ell_0 - 1} q_0^{\ell_0 + 1} = q_0^{2\ell_0}$ options. So $|C| = |C_1| + |C_2| = q_0^{2\ell_0}(2 - 1/q_0) = 2(q-1)^{\ell_0}(1 - 1/(2q_0)) = 2(q-1)^{\ell/2}(1 - 1/(2\sqrt{q-1}))$, which proves the cardinality given for C is correct.

For proving that the code is frameproof against 2 colluders, assume that the code is not frameproof and that some codeword $\vec{y}$ can be generated by the codewords $\vec{x}_1, \vec{x}_2$ which are both not equal to $\vec{y}$. We first claim that codewords $\vec{a} \in C_1, \vec{b} \in C_2$ can match in at most $\ell_0 - 1$ positions. For proving this, suppose $\vec{a} \in C_1, \vec{b} \in C_2$ match in ℓ_0 positions. Since their first coordinates differ, this implies that on the remaining positions there are ℓ_0 matches. This then implies that the polynomials f and s match on ℓ_0 positions. Since the degrees of both polynomials are less than ℓ_0, this means that $f = s$. But f has degree $\ell_0 - 1$ and s has degree stricly less than $\ell_0 - 1$, which gives a contradiction. (Note that here we make use of the fact that $\deg(f) = \ell_0 - 1$ instead of $\deg(f) \leq \ell_0 - 1$.)

Now suppose $\vec{y} \in C_1$. Besides the first coordinate, there are $\ell - 1 = 2\ell_0 - 1$ positions, so that one of $\vec{x}_1, \vec{x}_2$ must match with $\vec{x}$ on at least ℓ_0 positions. Hence, by the previous result, if we assume w.l.o.g. that $\vec{x}_1$ is this codeword, then $\vec{x}_1 \in C_1$, since any $\vec{a} \in C_1, \vec{b} \in C_2$ can match in at most $\ell_0 - 1$ positions. But then it follows that the polynomials f, g associated to $\vec{y}$ are the same as those associated to $\vec{x}_1$, so that $\vec{y} = \vec{x}_1$, which is a contradiction.

Suppose on the other hand that $\vec{y} \in C_2$. Then by similar reasoning, one of $\vec{x}_1, \vec{x}_2$ must match $\vec{y}$ on at least $\ell_0 + 1$ positions, which would imply $\vec{y} = \vec{x}_1$ (or $\vec{y} = \vec{x}_2$). This again gives a contradiction, which concludes the proof. □

	∞	$(f,g)(0)$	$(f,g)(1)$	$(f,g)(2)$		
Alice	25	0	5	10	$(f(X) = X)$	$(g(X) = 0)$
Bob	25	5	10	15	$(f(X) = X + 1)$	$(g(X) = 0)$
Charlie	25	10	15	20	$(f(X) = X + 2)$	$(g(X) = 0)$
Dave	25	15	20	0	$(f(X) = X + 3)$	$(g(X) = 0)$
Eve	25	20	0	5	$(f(X) = X + 4)$	$(g(X) = 0)$
Fred	25	0	10	20	$(f(X) = 2X)$	$(g(X) = 0)$
George	25	5	15	0	$(f(X) = 2X + 1)$	$(g(X) = 0)$
...	...	...	...	...	...	...
Zoey (500)	25	24	18	12	$(f(X) = 4X + 4)$	$(g(X) = 4X + 4)$

Table 4.4: *The 2-frameproof code $\mathcal{C}_1$ from Theorem 4.17, with $\ell_0 = 2, \ell = 4, q_0 = 5$ and $q = 26$. We use the mapping $(i,j) \mapsto 5i + j$ to map points $(f(\alpha), g(\alpha))$ to symbols from an alphabet of size 26, and we map ∞ to symbol 25. Codewords from $\mathcal{C}_1$ are formed by taking polynomials $f(X) = aX + b$ and $g(X) = cX + d$, with $a, b, c, d \in \mathbb{F}_5$ and $a \neq 0$, giving $|\mathcal{C}_1| = 500$.*

	$t(3,4)$	$(s,t)(0)$	$(s,t)(1)$	$(s,t)(2)$		
Adam	0	0	0	0	$(s(X) = 0)$	$(t(X) = 0)$
Ben	0	5	5	5	$(s(X) = 1)$	$(t(X) = 0)$
Chris	0	10	10	10	$(s(X) = 2)$	$(t(X) = 0)$
Donald	0	15	15	15	$(s(X) = 3)$	$(t(X) = 0)$
Eleanor	0	20	20	20	$(s(X) = 4)$	$(t(X) = 0)$
Fiona	6	1	1	1	$(s(X) = 0)$	$(t(X) = 1)$
Greg	6	6	6	6	$(s(X) = 1)$	$(t(X) = 1)$
...	...	...	...	...	...	...
Zora (625)	14	24	22	23	$(s(X) = 4)$	$(t(X) = 4X^2 + 4X + 4)$

Table 4.5: *The 2-frameproof code $\mathcal{C}_2$ from Theorem 4.17, with $\ell_0 = 2, \ell = 4, q_0 = 5$ and $q = 26$. Codewords from $\mathcal{C}_2$ are formed by taking polynomials $s(X) = a$ and $t(X) = bX^2 + cX + d$, with $a, b, c, d \in \mathbb{F}_5$, giving $|\mathcal{C}_2| = 625$. In total the code $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$ has cardinality $n = 1125 = q_0^{\ell-1}(2q_0 - 1)$.*

In the same paper of Theorem 4.12, Blackburn investigated whether an even stronger bound could be derived, finally resulting in the following Theorem. This indeed gives a stronger bound in some cases. We will not go into the details of the proof here, as it would take several pages to prove the result while not adding value to the report. We will only give a sketch of the proof.

Theorem 4.18. *[Bla03c, Theorem 11, Corollary 12] Let t, k_1, ℓ be positive integers such that $1 \leq k \leq \ell$ and $t \in \{1, \ldots, c\}$, such that $k_1 = \lceil \ell/c \rceil$ and $t \equiv \ell \mod c$. Then*

$$F(c, \ell, q) \leq \left(\frac{\ell}{\ell - (t-1)k_1} \right) q^{k_1} + \binom{\ell}{k_1 - 1} q^{k_1 - 1} = \left(\frac{\ell}{\ell - (t-1)k_1} \right) q^{k_1} + \mathcal{O}(q^{k_1 - 1}). \quad (4.6)$$

Sketch of the proof. The proof in [Bla03c] goes via what is introduced as t-colliding sets (a collection of subsets such that no t sets are disjoint) and frameproof code set systems. First it is proven in [Bla03c, Lemma 7] that if a frameproof code of cardinality $|C|$ exists, then a frameproof code set system of cardinality $|D|$ exists such that $|D| \geq |C| - \binom{\ell}{k-1} q^{k-1}$. Then in Theorem 8 [Bla03c] a bound on $|D|$ is given, namely $|D| \leq q^k / (1 - m(t, k, \ell)/\binom{\ell}{k}) = \kappa q^k$. Finally [Bla03c, Theorem 11] gives an upper bound on $m(t, k, \ell)$, which combined with the above gives the desired result. □

For a specific choice of parameters, namely $\ell = 5$ and $c = 3$, Blackburn included a construction in [Bla03c] showing that this bound is also at least in some cases (almost) tight. For $\ell = 5, c = 3$ we get $k_1 = 2$, $t = 2$ so that any q-ary code has cardinality at most $\frac{5}{3}q^2 + o(q)$ according to Theorem 4.18. This is indeed a stronger bound than the one from Theorem 4.12, which in this case gives $n \leq \max\{q^2, 2(q^2 - 1) + (q - 1)\} = 2q^2 + \mathcal{O}(q)$. The following 3-frameproof code has cardinality $\frac{5}{3}q^2 - o(q^2)$, thus showing that in this case Blackburn's second bound is tight up to order terms. We will not give a proof that this code is indeed 3-frameproof, as again the proof would only be lengthy and not add any value to this report. For a proof, see [Bla03c].

Lemma 4.19. *[Bla03c, Construction 4] Let $q = 3q_0 + 1$, with $q_0 \geq 4$ a prime power. Let the sets $X_1, \ldots, X_5 \subseteq (\mathbb{Z}_3 \cup \{\infty\})^5$ of size 3 each be defined as*

$$X_1 = \{(\infty, a, a, a, a) \mid a \in \mathbb{Z}_3\} \quad (4.7)$$
$$X_2 = \{(a, \infty, a, a+1, a+2) \mid a \in \mathbb{Z}_3\} \quad (4.8)$$
$$X_3 = \{(a, a, \infty, a+2, a+1) \mid a \in \mathbb{Z}_3\} \quad (4.9)$$
$$X_4 = \{(a, a+1, a+2, \infty, a) \mid a \in \mathbb{Z}_3\} \quad (4.10)$$
$$X_5 = \{(a, a+2, a+1, a, \infty) \mid a \in \mathbb{Z}_3\} \quad (4.11)$$

Let $\alpha_1, \ldots, \alpha_4$ be distinct elements from $\mathbb{Z}_{q_0}$, and let the sets $Y_1, \ldots, Y_5 \subseteq (\mathbb{Z}_{q_0} \cup \{\infty\})^5$ of cardinality q_0^2 each be defined as

$$Y_1 = \{(\infty, f(\alpha_1), f(\alpha_2), f(\alpha_3), f(\alpha_4)) \mid f \in \mathbb{Z}_{q_0}[X], \deg(f) \leq 1\} \quad (4.12)$$
$$Y_2 = \{(f(\alpha_1), \infty, f(\alpha_2), f(\alpha_3), f(\alpha_4)) \mid f \in \mathbb{Z}_{q_0}[X], \deg(f) \leq 1\} \quad (4.13)$$
$$Y_3 = \{(f(\alpha_1), f(\alpha_2), \infty, f(\alpha_3), f(\alpha_4)) \mid f \in \mathbb{Z}_{q_0}[X], \deg(f) \leq 1\} \quad (4.14)$$
$$Y_4 = \{(f(\alpha_1), f(\alpha_2), f(\alpha_3), \infty, f(\alpha_4)) \mid f \in \mathbb{Z}_{q_0}[X], \deg(f) \leq 1\} \quad (4.15)$$
$$Y_5 = \{(f(\alpha_1), f(\alpha_2), f(\alpha_3), f(\alpha_4), \infty) \mid f \in \mathbb{Z}_{q_0}[X], \deg(f) \leq 1\} \quad (4.16)$$

Let the codes $\mathcal{C}_1, \ldots, \mathcal{C}_5$ of cardinality $3q_0^2$ each be defined as

$$\mathcal{C}_1 = \{((x_1, y_1), \ldots, (x_5, y_5)) \mid (x_1, \ldots, x_5) \in X_1, (y_1, \ldots, y_5) \in Y_1\} \quad (4.17)$$
$$\mathcal{C}_2 = \{((x_1, y_1), \ldots, (x_5, y_5)) \mid (x_1, \ldots, x_5) \in X_2, (y_1, \ldots, y_5) \in Y_2\} \quad (4.18)$$
$$\mathcal{C}_3 = \{((x_1, y_1), \ldots, (x_5, y_5)) \mid (x_1, \ldots, x_5) \in X_3, (y_1, \ldots, y_5) \in Y_3\} \quad (4.19)$$
$$\mathcal{C}_4 = \{((x_1, y_1), \ldots, (x_5, y_5)) \mid (x_1, \ldots, x_5) \in X_4, (y_1, \ldots, y_5) \in Y_4\} \quad (4.20)$$
$$\mathcal{C}_5 = \{((x_1, y_1), \ldots, (x_5, y_5)) \mid (x_1, \ldots, x_5) \in X_5, (y_1, \ldots, y_5) \in Y_5\} \quad (4.21)$$

Then the code $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{C}_3 \cup \mathcal{C}_4 \cup \mathcal{C}_5$ *is 3-frameproof of length* 5 *and cardinality* $15q_0^2 = \frac{5}{3}(q-1)^2 = \frac{5}{3}q^2 - \frac{10}{3}q + \frac{5}{3}$.

	(∞,∞)	$(a, f(0))$	$(a, f(1))$	$(a, f(2))$	$(a, f(3))$	
Alice	15	0	0	0	0	$(f(X) = 0)$
Bob	15	1	1	1	1	$(f(X) = 1)$
Charlie	15	2	2	2	2	$(f(X) = 2)$
Dave	15	3	3	3	3	$(f(X) = 3)$
Eve	15	4	4	4	4	$(f(X) = 4)$
Fred	15	0	1	2	3	$(f(X) = X)$
George	15	1	2	3	0	$(f(X) = X + 1)$
...	...	...	...	...	...	...
Zoey (75)	15	14	13	12	11	$(f(X) = 4X + 4)$

Table 4.6: *The* 2*-frameproof code* $\mathcal{C}_1$ *from Lemma 4.19, with* $\ell = 5, m = 5$ *and* $q = 16$*. We use the mapping* $(i, j) \mapsto 5i + j$ *to map points* $(a, f(\alpha))$ *to symbols from an alphabet of size* 16*, and we map the pair* (∞, ∞) *to symbol* 15*.*

	$(a, f(0))$	$(a+2, f(1))$	$(a+1, f(2))$	$(a, f(3))$	(∞,∞)	
Adam	0	10	5	0	15	$(f(X) = 0)$
Ben	1	11	6	1	15	$(f(X) = 1)$
Chris	2	12	7	2	15	$(f(X) = 2)$
Donald	3	13	8	3	15	$(f(X) = 3)$
Eleanor	4	14	9	4	15	$(f(X) = 4)$
Fiona	0	11	7	3	15	$(f(X) = X)$
Greg	1	12	8	4	15	$(f(X) = X + 1)$
...	...	...	...	...	...	...
Zora (75)	14	13	12	11	15	$(f(X) = 4X + 4)$

Table 4.7: *The* 2*-frameproof code* $\mathcal{C}_5$ *from Lemma 4.19, with* $\ell = 5, m = 5$ *and* $q = 16$*.*

Finally, before we move on to the relation with error-correcting codes, we give a result from [SW98] which also shows that for large q, one can get codelengths of $\ell = \mathcal{O}(c^2 \log(n))$.

Lemma 4.20. *[SW98, Theorem 3.11] For any prime power* q *and any integer* $t < q$ *there exists a* $\lfloor q/(t-1) \rfloor$*-frameproof code of length* $q^2 + q$ *and cardinality* q^t*.*

4.2.1 Constructions from linear error-correcting codes

As mentioned earlier, there is a strong relationship between several classes of deterministic static schemes and error-correcting codes. Frameproof codes can also be constructed from certain error-correcting codes, as we will show here. First of all, the following result characterizes which linear error-correcting codes are 2-frameproof codes.

Lemma 4.21 (2-frameproof codes from linear error-correcting codes)**.** *[CE00, Theorem 1] A linear* $[\ell, k, d]_q$ *error-correcting code is* 2*-frameproof if and only if* $\mathrm{supp}(\vec{x}_1) \cap \mathrm{supp}(\vec{x}_2) \neq \emptyset$ *for all* $\vec{x}_1, \vec{x}_2 \in \mathcal{C}$*.*

Proof. Suppose $\mathrm{supp}(\vec{x}_1) \cap \mathrm{supp}(\vec{x}_2) = \emptyset$ for some $\vec{x}_1, \vec{x}_2 \in \mathcal{C}$. Then $\vec{x}_1 + \vec{x}_2 \in \mathcal{C}$ since $\mathcal{C}$ is linear, and entries of $\vec{x}_1 + \vec{x}_2$ satisfy $0 \leq (\vec{x}_1 + \vec{x}_2)_i \leq q - 1$ for all $1 \leq i \leq \ell$. Since $\vec{x}_1 + \vec{x}_2 \in \langle \vec{x}_1, \vec{x}_2 \rangle$, the code is not 2-frameproof.

For the converse, let $\vec{x}_1, \vec{x}_2, \vec{y}$ be three distinct codewords. Then $\vec{y} - \vec{x}_1$ and $\vec{y} - \vec{x}_2$, where subtraction is done modulo q, have a non-empty intersecting support, i.e. there exists some position i such that $(\vec{y}-\vec{x}_1)_i \neq 0$ and $(\vec{y}-\vec{x}_2)_i \neq 0$. This means that $(\vec{y})_i \neq (\vec{x}_1)_i$ and $(\vec{y})_i \neq (\vec{x}_2)_i$, hence the code is 2-frameproof. □

One can prove that any linear error-correcting code with sufficiently large minimum distance can be used as a c-frameproof code, where c depends on ℓ and d only, as the following Lemma shows. However, for most codes k is too large, i.e. the code contains too many codewords, so that the code will only be (trivially) 1-frameproof and not even 2-frameproof.

Lemma 4.22 (Frameproof codes from linear error-correcting codes)**.** *[CE00, Proposition 1] A linear $[\ell, k, d]$ error-correcting code is $\lceil d/(\ell - d)\rceil$-frameproof.*

Proof. Suppose $\vec{x}_1, \ldots, \vec{x}_c, \vec{y} \in \mathcal{C}$ and $\vec{y} \in \langle \vec{x}_1, \ldots, \vec{x}_c \rangle$. Since $\vec{y}$ matches with $\vec{x}_1, \ldots, \vec{x}_c$ on a total of ℓ positions, $\vec{y}$ must match with at least one of those $\vec{x}_j$ on at least $\lceil \ell/c \rceil$ positions. Since any two codewords can only match on at most $\ell - d$ coordinates, it follows that only if $c \geq \lceil \ell/(\ell-d) \rceil$, an innocent user can be framed. Hence taking $c \leq \lceil \ell/(\ell-d) \rceil - 1 = \lceil d/(\ell-d) \rceil$ assures that no innocent user can be framed. □

Corollary 4.23. *If a linear $[\ell, k, d]$ error-correcting code $\mathcal{C}$ satisfies $d \geq \ell(1 - 1/c)$, then $\mathcal{C}$ is c-frameproof.*

If we distinguish between MDS and non-MDS codes, we can even get exact values for the maximum value of c for which a linear error-correcting code is c-frameproof, as the following results show.

Lemma 4.24 (Frameproof codes from non-MDS error-correcting codes)**.** *[CE00, Proposition 2] An $[\ell, k, d]$ non-MDS-code $\mathcal{C}$ is at most $\lceil (\ell - d' + 1)/(d' - 1) \rceil$-frameproof, where d' is the minimum distance of the dual code with parameters $[\ell, \ell - k, d']$.*

Proof. From coding theory we know that for any $d' - 1$ positions and any $d' - 1$ values assigned to those positions, there are exactly $q^{k-d'+1} > 1$ codewords taking those values on those coordinates. Hence we can partition $[\ell]$ in $\lceil \ell/(d'-1) \rceil$ sets of at most $d' - 1$ positions, and find codewords $\vec{x}_j$ for each set such that $\vec{x}_j$ and $\vec{y}$ have the same symbols on those positions. Therefore one can frame any user using a coalition of size $\lceil \ell/(d'-1) \rceil$, so that $\mathcal{C}$ is at most $\lceil \ell/(d'-1) \rceil - 1 = \lceil (\ell - d' + 1)/(d'-1) \rceil$-frameproof. □

Lemma 4.25 (Frameproof codes from MDS error-correcting codes)**.** *[CE00, Corollary 1] An $[\ell, k, d]$ MDS-code $\mathcal{C}$ is exactly $\lceil d/(\ell - d) \rceil$-frameproof.*

Proof. If $\mathcal{C}$ is MDS, then $d' = k + 1$. Therefore the proof of the previous lemma, to prove that $\mathcal{C}$ is at most $\lceil (\ell - d' + 1)/(d' - 1) \rceil$-frameproof, does not work anymore, because $q^{k-d'+1} = 1$, hence $\vec{y}$ is the only word with those symbols on those $d' - 1$ positions. However, we can use the same method to prove that $\mathcal{C}$ is at most $\lceil (\ell - d' + 2)/(d' - 2) \rceil$-frameproof. Since for MDS codes, $d' = k + 1$, we get that $\mathcal{C}$ is at most $\lceil (\ell - d' + 2)/(d' - 2) \rceil = \lceil (\ell - k + 1)/(k - 1) \rceil = \lceil d/(\ell - d) \rceil$-frameproof, while we saw earlier that $\mathcal{C}$ is at least $\lceil d/(\ell - d) \rceil$-frameproof. So $\mathcal{C}$ is exactly $\lceil d/(\ell - d) \rceil$-frameproof. □

Example 4.26 (Frameproof codes from MDS error-correcting codes)**.** *[CE00, Example 1] A $[c + 1, 2, c]$ MDS-code over an alphabet of size $c + 1$ is a c-frameproof code of length $c + 1$ and cardinality q^2.*

Boneh and Shaw gave the following result regarding the existence of error-correcting codes with large minimum distance, so that one can also construct frameproof codes from these codes.

Lemma 4.27. *[BS98, Lemma III.3] Let $\ell = 8c\log(n)$. Then there exists an error-correcting code of length ℓ, cardinality n and alphabet size $2c$ with minimum distance $d > \ell(1 - 1/c)$.*

Corollary 4.28. *Let $\ell = 8c\log(n)$. Then there exists a c-frameproof code of length ℓ, cardinality n and alphabet size $2c$.*

This shows that for an alphabet of size $2c$ we can construct frameproof codes of length $\mathcal{O}(c\log(n))$. However, we prefer to have codes with smaller alphabets. Using code concatenation, described in the next subsection, we can reduce the alphabet size, at the cost of increasing the codelength.

4.2.2 Concatenating codes

Finally, we conclude the section on frameproof codes by a method to obtain frameproof codes with new, different parameters from other frameproof codes which we know to exist. This is done through concatenation of codes. Below is first the definition of concatenated codes, followed by its application to frameproof codes.

Definition 4.29 (Concatenating codes)**.** *For $i = 1, 2$, let $\mathcal{C}_i$ be a code of length ℓ_i, cardinality n_i over an alphabet of size q_i, and let $n_1 > q_2$. Then the* ***concatenated code*** *$\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ is the code obtained by replacing the q_2 distinct symbols in $\mathcal{C}_2$ by distinct codewords from $\mathcal{C}_1$. This concatenated code has length $\ell_1\ell_2$, cardinality n_2 and uses an alphabet of size q_1.*

Lemma 4.30 (Concatenating frameproof codes)**.** *[CE00, Proposition 5] [BS98, Lemma III.2 for $q_1 = 2$] Let $\mathcal{C}_1, \mathcal{C}_2$ be c-frameproof codes of length ℓ_i, cardinality n_i over an alphabet of size q_i, for $i = 1, 2$, and let $n_1 > q_2$. Then the code $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ is a c-frameproof code.*

Proof. Because $\mathcal{C}_2$ is c-frameproof, one cannot generate $\vec{y} \in \mathcal{C}_2$ from at most c other codewords in $\mathcal{C}_2$. In other words, for each such tuple of $c + 1$ codewords, there exists some position i such that $(\vec{y})_i \notin \{(\vec{x}_1)_i, \ldots, (\vec{x}_c)_i\}$. Replacing symbols by codewords, we see that this expression becomes $(\vec{y})_i = \vec{y}' \notin \{\vec{x}'_1, \ldots, \vec{x}'_c\} = \{(\vec{x}_1)_i, \ldots, (\vec{x}_c)_i\}$, for some codewords $\vec{y}', \vec{x}'_1, \ldots, \vec{x}'_c \in \mathcal{C}_1$. This condition is guaranteed by the c-frameproofness of $\mathcal{C}_1$, which proves the result. □

Let us illustrate this method through an example, in which we concatenate a binary 2-frameproof code of cardinality 4 and a quaternary 2-frameproof code of cardinality 6 to form a binary 2-frameproof code of cardinality 6.

Example 4.31. *Let $\mathcal{C}_1 = \{000, 011, 101, 110\}$ be a binary 2-frameproof code of length $\ell_1 = 3$ and cardinality $n_1 = 4$, and let $\mathcal{C}_2 = \{10, 20, 30, 01, 02, 03\}$ be a quaternary 2-frameproof code of length $\ell_2 = 2$ and cardinality $n_2 = 6$. Then the code $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2 = \{011000, 101000, 110000, 000011, 000101, 000110\}$, also given in Table 4.8, is a binary 2-frameproof code of length $\ell_1\ell_2 = 6$ and cardinality $n_2 = 6$.*

The concatenation of codes can be used to construct families of codes with increasing ℓ, maintaining a somewhat good efficiency. The following result is also a consequence of repetitive code concatenations, giving a family of 2-frameproof codes with a length polynomial in $\log(n)$.

Lemma 4.32. *[SW98, Theorem 3.15] For any $j \geq 1$, there exists a binary 2-frameproof code of length $6 \cdot 4^j$ and cardinality 5^{2^j}.*

	block 1			block 2		
Alice	0	1	1	0	0	0
Bob	1	0	1	0	0	0
Charlie	1	1	0	0	0	0
Dave	0	0	0	0	1	1
Eve	0	0	0	1	0	1
Fred	0	0	0	1	1	0

Table 4.8: *The 2-frameproof code $\mathcal{C}_1 \circ \mathcal{C}_2$ for $n_2 = 6$ users with length $\ell_1\ell_2 = 6$.*

Note that the code concatenation can be used to turn any non-binary frameproof code into a binary one, by concatenating it with a sufficiently large binary frameproof code. So we can also turn the frameproof codes obtained in Corollary 4.28 into binary frameproof codes, as the following Theorem shows.

Theorem 4.33. *[BS98, Theorem III.4] Let $\ell = 16c^2 \log(n)$. Then there exists a binary c-frameproof code of length ℓ and cardinality n.*

Proof. Let $\mathcal{C}_1 = \Gamma_2(2c)$ and let $\mathcal{C}_2$ be the c-frameproof code of cardinality n from Lemma 4.27. Then $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ is a binary c-frameproof code of length $\ell_1\ell_2 = 16c^2 \log(n)$ and cardinality n. □

This last construction is quite a good construction: it uses a small alphabet, it can be used for any number of colluders, and it has a relatively short codelength. Also, the upper bounds on the cardinality of a frameproof code give $n \leq q^{\mathcal{O}(\ell/c)}$ while this construction gives $n = q^{\mathcal{O}(\ell/c^2)}$. It is still a factor c off in the exponent of q, but it is quite close to the upper bound for not too large c.

4.3 Secure frameproof codes

First of all, let us consider the traceability of secure frameproof codes; how well can we accuse traitors, using secure frameproof codes? For the case $c = 2$, as we saw in the Introduction, we get the following result.

Theorem 4.34 (Traceability of 2-secure frameproof codes)**.** *[STW00, Theorem 2.3] Let $\mathcal{C}$ be a 2-secure frameproof code, and let $\vec{y}$ be a forgery generated by some coalition C. Then either (1) at least one guilty user can be identified, or (2) a set of three users can be identified, two of which must be guilty.*

Proof. Suppose we are given such a code and a forgery. One can then check for all coalitions of size (at most) two whether they could have generated $\vec{y}$. Let G be this set of coalitions. Obviously $|G| \geq 1$, while if $|G| = 1$ (all members of) this coalition must be guilty. If $|G| = 2$, then since $\mathcal{C}$ is 2-secure frameproof, these two coalitions contain a common user, which must be guilty. For $|G| \geq 3$, suppose that no single user can be caught, i.e. there is no common member of all coalitions. Then, since each pair still has one user in common, the first three coalitions have users say $\{1, 2\}, \{1, 3\}, \{2, 3\}$. Since any other coalition must have a user in common with all these three coalitions, all other coalitions are also a subset of $\{1, 2, 3\}$. So this set $\{1, 2, 3\}$ can be identified, which must contain two traitors. □

The above problem, of finding a subset of all the users with a rate of traitors as high as possible, can be formulated in terms of pairwise intersecting sets as follows.

Definition 4.35. *Let $U = \{1, \ldots, n\}$ and let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of subsets of U, such that $|S_i| = t$ for each i, and $|S_i \cap S_j| \geq 1$ for each i, j. Then the set $\mathcal{S}$ is called a **t-pairwise intersecting set**. Furthermore, let k be fixed and unknown and let $S_k \subseteq U$ be the **kernel** of U. The problem is to find a subset $W \subseteq U$ such that W contains as many elements from S_k as possible, and as few elements from $U \setminus S_k$ as possible. Writing $R_{S_k}(W) = |W \cap S_k|/|W|$ and $R = \max_{W \subseteq U} \min_{k \in [m]} R_{S_k}(W)$, the problem is to find R and preferably a W achieving this rate R.*

This problem relates to the secure frameproof codes in the following way. Take U to be the set of users, and let $\mathcal{S} = \mathcal{S}(\vec{y})$ contain those coalitions of size (at most) c that could have generated some obtained forgery $\vec{y}$. Then the set $\mathcal{S}$ is a c-pairwise intersecting set. We now take as the kernel the unknown set of real colluders C that generated $\vec{y}$. The problem is to find a subset of the users that contains as many guilty and as few innocent users as possible.

The above problem was solved for $c = 2$, since then one can either find a subset K' of size 1 containing 1 traitor or of size 3 containing at least 2 traitors (i.e. the rate of guilty users is either 1 or 2/3). For $c \geq 3$ we conjecture the following result, which seems to be new, but also seems hard to prove.

Conjecture 4.36. *There exist c-secure frameproof codes $\mathcal{C}$ which achieve no better rate than $\frac{c}{c^2-c+1} = \frac{1}{c} + \mathcal{O}(\frac{1}{c^2})$. In particular, if $\mathcal{C}$ is a code that is only known to be c-secure frameproof, then one generally cannot achieve a better success rate than $\frac{1}{c}(1 + 1/c)$.*

Evidence. We will give evidence that there exists a c-pairwise intersecting set such that the highest rate achievable is probably $c/(c^2 - c + 1)$. As we make no assumptions on the structure of the code, the collection of coalitions that could have generated a forgery could be anything. Thus we can choose it any way we want. This then proves that there exists a c-secure frameproof code for which one cannot necessarily do better than $\frac{1}{c}(1 + 1/c)$.

Consider the projective plane of order $c-1$, and identify the points as users and lines as coalitions of users. Then all lines (coalitions) contain c points, and any two lines intersect in exactly one point. If the coalitions formed by these lines were indeed exactly the coalitions able to produce some forgery $\vec{y}$, then we have a system of $(c-1)^2 + (c-1) + 1 = c^2 - c + 1$ users, c of which are guilty. Also we conjecture that there is no smaller subset of points/lines known to contain a higher rate of traitors than this whole projective plane; removing one point could reduce the number of traitors by 1 (two lines go through this point, i.e. if one of these lines is the line corresponding to the actual coalition, then the number of traitors is reduced by one), removing two points could reduce the number of traitors by 2 (exactly one line goes through any two points, i.e. if this line were the coalition, two colluders would be removed), and analogously, removing k points such that $\binom{k}{2} > c + 1$ could reduce the number of traitors by 3 (at least one line contains at least 3 of these points), et cetera. Therefore the highest rate achievable for this c-pairwise intersecting set is conjectured to be $c/(c^2 - c + 1)$. □

The above of course does not imply that any c-secure frameproof code has bad traceability, as for instance c-IPP codes are also c-secure frameproof codes. However, if we do not know more about this code than that it is c-secure frameproof, then we cannot draw any better conclusions than that we can get an accusation rate of $\mathcal{O}(1/c)$ as c tends to infinity.

Similar to the frameproof codes, we can get the following results about error-correcting codes and code concatenation for secure frameproof codes. In this case only a very specific class

Alice	1	1	1	0	0	0	0
Bob	1	0	0	1	1	0	0
Charlie	1	0	0	0	0	1	1
Dave	0	1	0	1	0	0	1
Eve	0	1	0	0	1	1	0
Fred	0	0	1	0	1	0	1
George	0	0	1	1	0	1	0
Forgery	1	1	1	1	1	1	1

Table 4.9: *The 3-secure frameproof code based on the Fano plane. Each column/row represents a line in the Fano plane. One can verify that the only 7 size-3 coalitions that could have generated the forgery are exactly the 7 lines from the Fano plane, and since all lines in the Fano plane intersect, it follows that the code is 3-secure frameproof. However, given the forgery above, the distributor cannot accuse a subset of users $V \subseteq U$ such that $|V \cap C|/|V| > \frac{3}{7}$ for any possible coalition C.*

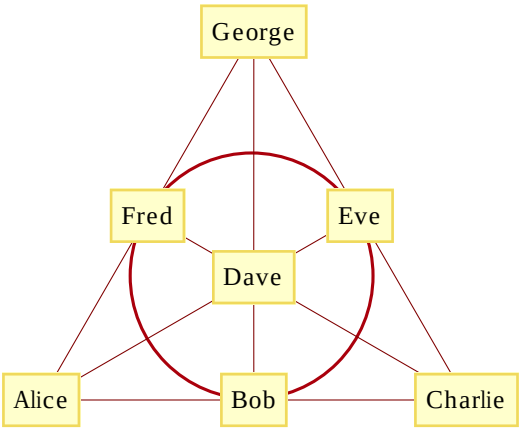

Figure 4.1: *The Fano plane, i.e. the projective plane over $\mathbb{F}_2$ consisting of 7 lines (going through 3 points each) and 7 points. Here we associate points with users, and lines with possible coalitions of size (at most) 3 that could have generated the forgery $\vec{y} = (1, 1, 1, 1, 1, 1, 1)$. The best achievable rate of guilty users is obtained by accusing the set of all 7 users. Then at least 3 of them are guilty, giving a success rate of $\frac{3}{7}$.*

of error-correcting codes translates to secure frameproof codes, as can be seen below, while concatenation of two secure frameproof codes again gives a secure frameproof code.

Theorem 4.37. *[EC01, Proposition 1] A binary constant weight linear error-correcting code with parameters $[\ell = 4i - 1, k, d = 2i]$ satisfying that any two distinct codewords agree in exactly $\lfloor d/2 \rfloor$ positions, is a 2-secure frameproof code of length ℓ and cardinality 2^k.*

Theorem 4.38. *[EC01, Section 5] Let $\mathcal{C}_1$ be a 2-secure frameproof code of length ℓ_1, cardinality n_1 over an alphabet of size q_1, and let $\mathcal{C}_2$ be a 2-secure frameproof code of length ℓ_2, cardinality n_2 over an alphabet of size q_2, and let $n_1 > q_2$. Then $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ is a 2-secure frameproof code of length $\ell_1\ell_2$, cardinality n_1 over an alphabet of size q_1.*

As for lower bounds, we only mention the following results, which hold for general c and q.

Theorem 4.39. *[SZ08, Theorem 7] If a c-secure frameproof code of length ℓ exists, then $n \leq q^{\lceil \ell/c \rceil} + 2c - 2$ (i.e. $\ell = \Omega(c \log_q(n))$).*

Theorem 4.40. *[SZ08, Corollary 14] If a c-secure frameproof code of length ℓ exists, then $n \leq (2c^2 - 3c + 2)q^{\lceil \ell/(2c-1) \rceil} - 2c^2 + 3c - 1$ (i.e. $\ell = \Omega(2c \log_q(n))$).*

4.4 IPP codes

Finally we turn our attention to the class of IPP codes, which are the weakest deterministic static schemes to have full traceability. As mentioned in the Introduction, the jump from weak or no traceability to being able to trace at least one user with certainty carries a big disadvantage, which is stated and now also proven below.

Theorem 4.41. *Let $\mathcal{C}$ be a c-IPP code with alphabet size q and cardinality $n > c$. Then $q > c$.*

Proof. Let $\vec{x}_1, \ldots, \vec{x}_{c+1}$ be distinct codewords from $\mathcal{C}$. Since these are more codewords than there are symbols in the alphabet, on every position i there exists some symbol α_i that occurs at least twice among $\{(\vec{x}_1)_i, \ldots, (\vec{x}_{c+1})_i\}$. Let $\vec{y}$ be the codeword defined by $y_i = \alpha_i$. Then every subset of c users of these $c+1$ users could have generated $\vec{y}$, since on any position at least one of their codewords matches $\vec{y}$. Therefore any of these $c+1$ users could theoretically still be innocent, hence there exists no tracing algorithm σ such that both $\sigma(\vec{y}) \subseteq C$ and $\sigma(\vec{y}) \neq \emptyset$. □

The strongest and most important other Theorem known for IPP codes is the following, which gives lower and upper bounds for the cardinality of a c-IPP code with given ℓ and q. Similar to the section on frameproof codes, we use the notation $I(c, \ell, q)$ to denote the maximum cardinality of a c-IPP code of length ℓ and alphabet size q.

Theorem 4.42. *[AS04, Theorem 2.1] Let $k = \lfloor c^2/4 \rfloor + c = \Theta(c^2)$. Then there exist two functions f_1, f_2 of c such that for every ℓ and q, $[f_1(c)q]^{\ell/k} < I(c, \ell, q) < f_2(c)q^{\lceil \ell/k \rceil}$.*

Corollary 4.43. *Any c-IPP code satisfies $\ell = \Omega(c^2 \log_q(n))$.*

Most work in literature on IPP codes has been focused on calculating or estimating $I(c, \ell, q)$ for several values of c, ℓ and q. The first work, [HVLLT98], focused on $I(2, 3, q)$ and $I(2, 4, q)$. For $I(2, 3, q)$, it was first proven that $I(2, 3, q) \geq \frac{3q}{2} - \mathcal{O}(1)$ [HVLLT98, Example 2], which was strengthened to $I(2, 3, q) \geq 2q - \mathcal{O}(\sqrt{q})$ [HVLLT98, Example 3], and finally a construction was given proving $I(2, 3, q) \geq 3q - \mathcal{O}(\sqrt{q})$, which for completeness is given below.

Lemma 4.44 (Constructive lower bound for $I(2, 3, q)$)**.** *[HVLLT98, Example 4] Let $k = \max\{i \in \mathbb{N} : i(i+2) \leq q\} \approx \sqrt{q}$. Then $I(2, 3, q) \geq 3k^2 = 3q - \mathcal{O}(\sqrt{q})$.*

Proof. Let the alphabet be partitioned into three subsets $S = \{1, \ldots, k\}$ (small numbers), $M = \{k+1, \ldots, 2k\}$ (medium numbers) and $L = \{2k+1, \ldots, k^2+2k\}$ (large numbers). Then $|S| = |M| = k$ and $|L| = k^2$. Let the code C be the union of the following three codes:

$$C_1 = \{(s_1, s_2, ks_1 + (k + s_2)) \mid s_1, s_2 \in S\} \subseteq S \times S \times L \tag{4.22}$$

$$C_2 = \{(m_1, ks_1 + m_1, s_1) \mid s_1 \in S, m_1 \in M\} \subseteq M \times L \times S \tag{4.23}$$

$$C_3 = \{(k(m_1 - k) + m_2, m_1, m_2) \mid m_1, m_2 \in M\} \subseteq L \times M \times M \tag{4.24}$$

Each of the codes contains k^2 codewords, so C has $3k^2$ codewords, and obviously the three codes are disjoint. Now suppose $\vec{c}$ contains a large coordinate. Then this coordinate uniquely determines a traitor, since every large coordinate occurs exactly once on each of the three positions. If on the other hand $\vec{c}$ contains no large coordinates, then we know the two sets C_i where the traitors came from (e.g. if $\vec{c} \in S \times M \times M$, then the traitors are from C_1 and C_3), and for one of these two codes we know two coordinates (e.g. if $\vec{s} \in S \times M \times M$, then the last two coordinates came from C_3). These two coordinates together uniquely identify one of the traitors. □

	(S)	(S)	(L)		(M)	(L)	(S)		(L)	(M)	(M)
Alice	1	1	5	Eve	3	5	1	Isaac	5	3	3
Bob	1	2	6	Fred	3	7	2	John	6	3	4
Charlie	2	1	7	George	4	6	1	Kelly	7	4	3
Dave	2	2	0	Henry	4	0	2	Linda	0	4	4

Table 4.10: *The 2-IPP code from Theorem 4.44 of length 3. Here we took $q = 8$ so that $k = 2$ and $n = 12$. The alphabet is partitioned as $Q = S \cup M \cup L$ with $S = \{1, 2\}, M = \{3, 4\}, L = \{5, 6, 7, 8\}$, and for consistency of alphabets starting at 0 we map the symbol 8 to 0.*

As for upper bounds on $I(2, 3, q)$, the paper [HVLLT98] first proves that $I(2, 3, q) \leq q^2$ [HVLLT98, Equation (7)], while later on it shows that $I(2, 3, q) \leq 3q - 1$ [HVLLT98, Theorem 1], making the bound and the above construction almost tight.

For $c = 2$ and a fixed codelength of $\ell = 4$, [HVLLT98] gives a construction achieving a cardinality of $n = q\sqrt{q} - \mathcal{O}(1)$. Later, in [AFS01], a better construction was given with $n = \Omega(q^2)$, while simultaneously proving an upper bound of $n = \mathcal{O}(q^2)$ [AFS01, Theorem 2.5, Theorem 3.4].

For $c = 2$ and variable ℓ and q, Hollmann et al. gave a construction for prime powers $q \geq \ell - 1$ with $I(2, \ell, q) \geq q^{\lceil \ell/4 \rceil}$ [HVLLT98, Corollary 1], while a same order upper bound on the cardinality was also given as $I(2, \ell, q) \leq 3q^{\lceil \ell/4 \rceil}$ [HVLLT98, Theorem 5].

In the case of $c = 3$, some results were also obtained explicitly, with a construction of cardinality $n = 5q - o(q)$ in [AS04, Theorem 3.1] for $\ell = 5$, while for $\ell = 6$ an upper bound was given on the cardinality as $n = o(q^2)$ [AS04, Theorem 3.2].

Similar to the case of frameproof codes, the case $\ell \leq c$ was completely solved, as [AS04, Lemma 4.1] shows that $I(c, \ell, q) = q$ for those cases. The case $\ell = c + 1$ was also studied in the same paper, resulting in $I(c, c + 1, q) = q + \frac{2q}{2c-3} - o(q)$ [AS04, Theorem 4.2].

Finally, not surprisingly we also get results through code concatenation and by using error-correcting codes, as follows.

Theorem 4.45. *[HVLLT98, Theorem 2] Let $\mathcal{C}_1$ be a 2-IPP code of length ℓ_1, cardinality n_1 over an alphabet of size q_1, and let $\mathcal{C}_2$ be a 2-IPP code of length ℓ_2, cardinality n_2 over an alphabet of size q_2, and let $n_1 > q_2$. Then $\mathcal{C} = \mathcal{C}_1 \circ \mathcal{C}_2$ is a 2-IPP code of length $\ell_1\ell_2$, cardinality n_1 over an alphabet of size q_1. In other words, $I(2, \ell_1\ell_2, q) \geq I(2, \ell_1, I(2, \ell_2, q))$.*

4.5 Summary

In this chapter we investigated deterministic static schemes, which are in a sense the most intuitive schemes. With these schemes one tries to find a construction for a code such that after receiving the forgery, always at least one traitor can be identified. We started off with a weaker class of codes, which only guarantees that users cannot be framed. These frameproof codes have relatively short codelengths (linear in c, logarithmic in n) and require only small alphabet sizes (binary). A slightly stronger class is the class of secure frameproof codes, which are codes that have the added property that any coalition that could have possibly generated the forgery actually contains a traitor. These codes also require a codelength linear in c and logarithmic in n, and one can still use a binary alphabet for these codes. When making the jump from this partial traceability to full traceability codes, i.e. IPP codes, we saw that the minimum codelength increases by a factor c (quadratic in c), and, more importantly, we can no longer use small alphabets: the minimum alphabet size is then $c + 1$. Especially for large c this is a big downside to using IPP codes.

Chapter 5

Probabilistic static schemes

***Citations**: For writing this chapter, the following articles were used: [AT09], [BT08], [BS98], [FGC08], [FPF09], [HM09b], [HM09a], [KSCS07], [Ker10], [NFH+09], [PSS03], [Sch03], [Sch04], [Sch06], [Sch08], [SS10], [SVCT06], [SKC08], [SKSC09], [Tar03], [Tar09], [Tar10], [Yac01].*

5.1 Introduction

In the previous chapter we considered deterministic static schemes. We saw that if we ask for no traceability or only weak traceability, then we can construct codes with a binary alphabet. However, when switching from this weak traceability to complete, deterministic traceability, i.e. always being able to identify at least one colluder, we saw that the required alphabet size suddenly becomes $c+1$. So somewhere between weak traceability and full traceability, there is a jump from small alphabets to large alphabets.

In this chapter we consider schemes with traceability that is stronger than the traceability of frameproof or secure frameproof codes, but weaker than the traceability of IPP codes. The schemes we consider in this chapter are allowed to have a small error margin in the process of tracing traitors. Instead of requiring a code to make no error in the accusations, we may use an accusation algorithm that is allowed to make a mistake with probability at most ϵ, where the probability is calculated over all possible fingerprinting codes that could have been used. Since making a mistake can essentially mean two different things, we first define these two types of errors, regarding **soundness** and **completeness** of the scheme. Here we explicitly refer to the scheme as a pair $(\mathcal{C}, \sigma)$, where $\mathcal{C}$ is the fingerprinting code and σ is the tracing algorithm.

Definition 5.1 (Soundness, completeness)**.** *Let $(\mathcal{C}, \sigma)$ be a fingerprinting scheme and let $c \geq 2$ and $\epsilon_1, \epsilon_2 > 0$. Then we say the scheme $(\mathcal{C}, \sigma)$ is a **c-sound scheme with ϵ_1-error** (also **c-frameproof scheme with ϵ_1-error**, or the scheme has a **false positive probability** of at most ϵ_1 against c colluders, or if c is implicit we say the scheme is ϵ_1-sound) if, for any fixed coalition C and pirate strategy $\rho : \mathcal{C} \mapsto \vec{y}$, the probability over all possible codes $\mathcal{C}$ that the accusation algorithm σ accuses at least one innocent user is bounded from above by ϵ_1. Similarly, we say the scheme $(\mathcal{C}, \sigma)$ is a **c-complete scheme with ϵ_2-error**, (or the code has a **false negative probability** of at most ϵ_2 against c colluders, or if c is implicit we say the scheme is ϵ_2-complete) if, for any fixed coalition C and pirate strategy ρ, the probability over $\mathcal{C}$ that the accusation algorithm does not accuse any guilty users is bounded from above by ϵ_2.*

In other words, soundness with error ϵ_1 implies that with probability at least $1 - \epsilon_1$ no innocent user will be accused, while completeness with error ϵ_2 implies that with probability at least

$1 - \epsilon_2$, at least one of the guilty users will be accused. Note that these probabilities are taken for a fixed coalition C and fixed pirate strategy ρ, but not for fixed codewords assigned to users and not for a fixed forgery generated by a coalition.

Since we want to have schemes that satisfy both these requirements, we sometimes also say a scheme is c-secure with (ϵ_1, ϵ_2)-error to indicate that the scheme is ϵ_1-sound and ϵ_2-complete against c colluders.

The rest of this chapter is devoted to reviewing results from literature about c-secure schemes with (ϵ_1, ϵ_2)-error. First, in Section 5.2, we look at theoretical lower bounds on the codelength of any c-secure, (ϵ_1, ϵ_2)-error scheme. Then, in Section 5.3, we look at the **Boneh-Shaw scheme**, introduced in [BS98], and we show why this scheme is not optimal. In Section 5.4 we then look at the **Tardos scheme**, which (up to a constant factor) achieves the best known lower bound for the asymptotic case of $c \to \infty$. For this last scheme we also look at several improvements suggested in literature, which further reduce the codelength. Finally, in Section 5.5 we summarize the results from this chapter.

5.2 Lower bounds

First of all, we present lower bounds from literature on ℓ, given c, ϵ_1 and ϵ_2. A lot of work has been done in this area, by e.g. Boneh and Shaw in [BS98], Peikert et al. in [PSS03], Tardos in [Tar03], Amiri and Tardos in [AT09] and Huang and Moulin in [HM09b], [HM09a]. We will discuss the results in order of strength, with the weakest but easiest to prove bounds first.

5.2.1 Linear in c

The following Theorem, with a not so advanced proof, shows that the codelength has to be at least linear in the number of colluders, and logarithmic in $1/(c\epsilon_1)$, if $\epsilon_1 = \epsilon_2$.

Theorem 5.2. *[BS98, Theorem VI.1] Let $\epsilon_2 \geq \epsilon_1 > 0$, and let $c \geq 2$. If a binary fingerprinting scheme $(\mathcal{C}, \sigma)$ with codelength ℓ is ϵ_1-sound and ϵ_2-complete against up to c colluders, then $\ell \geq \frac{1}{2}(c-3)\ln(1/c\epsilon_1) = \Omega(c \ln(1/c\epsilon_1))$.*

Proof. We prove that for any assignment of codewords of length $\ell < \frac{1}{2}(c-3)\ln(1/c\epsilon_1)$ to $c+1$ users, we can construct a word $\vec{y}$ which could have been generated by any coalition of c of these users with probability at least ϵ_1. This then proves that any code of this length cannot be both ϵ_1-sound and ϵ_2-complete, since this word $\vec{y}$ cannot be traced back to a guilty user with a high enough success probability by any tracing algorithm.

Let $\vec{x}_1, \ldots, \vec{x}_{c+1}$ denote the $c+1$ codewords assigned to $c+1$ users. Let $B(k)$ be the set of fingerprint positions such that exactly k of the $c+1$ users have a 1 there. Then obviously $\sum_{k=0}^{c} |B(k)| = \ell$, hence $\sum_{k=2}^{c-2} |B(k) \cup B(k+1)| \leq 2\ell < (c-3)\ln(1/c\epsilon_1)$, so that for at least one $2 \leq k_0 \leq c-2$ we have $|B(k_0) \cup B(k_0+1)| \leq \log(1/c\epsilon_1)$. We now define the word $\vec{y}$ as $y_i = 0$ if on position i the $c+1$ users have at most k_0 ones, and $y_i = 1$ otherwise. We will now show that any coalition of c users can guess this word with probability at least ϵ_1.[1]

Let C be a coalition of c of these $c+1$ users. First of all, the coalition guesses k_0 by guessing uniformly at random between 2 and $c-2$. This guess is correct with probability $1/(c-3) > 1/c$. Now let i be a position that is detectable by the coalition. The coalition then knows that there are, say, k ones among their codewords, hence $i \in B(k) \cup B(k+1)$. If $k < k_0$, then $i \in B(k)$ for some $k \leq k_0$, so the coalition puts a 0 on these positions, which matches with y_i. Similarly,

[1] Note that this word $\vec{y}$ can indeed be generated by any coalition of size c, for which we need that $2 \leq k_0 \leq c-2$.

if $k > k_0$, then $i \notin B(k)$ for any $k \leq k_0$, so that the coalition can put a 1 there, knowing that it will match $\vec{y}$ on this position. Finally, if $k = k_0$, the coalition does not know whether they should choose $y_i = 0$ or $y_i = 1$. Then the coalition simply chooses y_i uniformly at random, so that $\mathbb{P}[y_i = 1] = \mathbb{P}[y_i = 0] = 1/2$. So with probability $(1/2)^{|B(k_0) \cup B(k_0+1)|} \geq (1/2)^{\ln(1/c\epsilon_1)} = c\epsilon_1$, all these bits are guessed correctly, so that the probability of both guessing k_0 right and then also guessing $\vec{y}$ right is at least ϵ_1, as claimed. □

While the above Theorem was quite easy to prove, the bound it provides is not tight, as we will see below.

5.2.2 Quadratic in c

As it turns out, one can improve upon the lower bound of Boneh and Shaw given above by a factor c. The following Theorem given by Peikert et al. in [PSS03] shows that the codelength is always at least quadratic in c. The small price we pay is that we need some obscure condition on the size of $\ln(1/\epsilon_1)$ for this Theorem to hold.

Theorem 5.3. *[PSS03, Theorem 5.1] Let $\mathcal{C}$ be a c-secure scheme with $\epsilon = \epsilon_1 = \epsilon_2$ error, and let $\ln(1/\epsilon)$ be sufficiently large. Then $\ell = \Omega(c^2 \ln(1/c\epsilon))$.*

Sketch of the proof. The proof given in [PSS03] first describes a universal strategy, to be used against any fingerprinting scheme. On any position i, let k be the number of ones a coalition of size c sees on that position. For $k \leq \lfloor c/2 \rfloor$ we now let $r_k = \beta k^2$ with β such that $r_{\lfloor c/2 \rfloor} = 1/2$, while for $\lfloor c/2 \rfloor < k \leq c$ we define $r_k = (1 - r_{c-k})$. The strategy is then to output a 1 on position i with probability r_k and a 0 with probability $1 - r_k$. Since $r_0 = 0, r_c = 1$ (marking assumption) and $0 \leq r_k \leq 1$ this is a feasible pirate strategy.

Using this strategy, Peikert et al. then go on to show that with sufficiently large probability, a coalition can generate *ideal target words*. These are words which are ideal for pirates, in the sense that for the distributor it is then impossible (or at least sufficiently hard) to trace the traitors. This is similar to the approach used in the proof of Boneh and Shaw's lower bound: one tries to find a word $\vec{y}$ and $c + 1$ users such that any c-subset of these users could have generated this word with sufficiently large probability. Then any of these users can be innocent with sufficiently large probability, hence no single user can be accused with high enough certainty. □

In the same year, another quadratic bound was given by Tardos in [Tar03]. This bound also loses the factor c inside the logarithm in both Boneh and Shaw's Theorem and Peikert's Theorem, and this Theorem does not need that ϵ_1 is sufficiently small. However, now we do need the small requirement that ϵ_1 is really smaller than ϵ_2/c. This last requirement is usually satisfied, since one usually allows more room for error in not catching pirates than in catching innocent users.

Theorem 5.4. *[Tar03, Theorem 4] Let $\epsilon_1, \epsilon_2 > 0$ such that $\epsilon_1 < (\epsilon_2/c)^a$ for some $a > 1$, and let $c \geq 3$. If a fingerprinting code $\mathcal{C}$ of length ℓ is ϵ_1-sound and ϵ_2-complete against c colluders, then $\ell = \Omega(c^2 \ln(n/\epsilon_1))$, where the factor in the big Omega is some constant depending solely on a.*

Sketch of the proof. Whether it is a coincidence or not, the proof given by Gabor Tardos from the Rényi Institute in Hungary uses the seldom used ("esoteric", to quote Tardos) measure of distance known as the Rényi divergence. Similar to the lower bounds above, Tardos first provides a uniformly used pirate strategy. Given that there are k ones among the c bits seen on position i, Tardos also defines biases r_k for choosing $y_i = 1$. Where Peikert et al. used $r_k = \beta k^2$, Tardos

chooses the strategy ρ by $\mathbb{P}[y_i = 1] = r_k = 3k^2 - 2k^3$. A similar strategy ρ' is then defined for when only $c-1$ of these c users collude.

Now suppose that some scheme manages to accuse one of the guilty users, say user 1, with sufficiently large probability. Then $\mathbb{P}[1 \in \sigma(\rho(X))] > \epsilon_2$. Now we consider the strategy ρ' deployed by the other $c-1$ users of the coalition. Then also we have $\mathbb{P}[1 \in \sigma(\rho'(X))] \leq \epsilon_1$, as user 1 is now an innocent user. Tardos then continues by calculating the Rényi divergence between the distributions of $(\rho(X), X, \sigma)$ and $(\rho'(X), X, \sigma)$. These are then shown to be too close to eachother for $\mathbb{P}[1 \in \sigma(\rho(X))] > \epsilon_2$ and $\mathbb{P}[1 \in \sigma(\rho'(X))] \leq \epsilon_1$ to hold if ℓ is too small or if the relation between ϵ_1 and ϵ_2 is not satisfied. This then finally proves the result. □

These quadratic bounds in c are asymptotically tight, as we will later see constructions matching this quadratic bound. However, the above Theorems do not care about the constants, e.g. we still do not know whether $\ell \geq c^2 \ln(n/\epsilon)$, or $\ell \geq 1000c^2 \ln(n/\epsilon)$, or $\ell \geq (1/1000)c^2 \ln(n/\epsilon)$. In the next subsection we will investigate results regarding this final constant.

5.2.3 Finding the final constant

Let d_ℓ^* denote the constant such that there exist schemes with length $\ell = d_\ell^* c^2 \ln(1/\epsilon_1)$ (for asymptotically large c) but there exist no schemes with a shorter codelength. In other words, d_ℓ^* is the critical constant for which secure schemes just exist. The paper [HM09b] investigated this d_ℓ^*, and concluded that $\ln(2) \leq d_\ell^*$ [HM09b, Theorem 4.2] and $d_\ell^* \leq \pi^2 \ln(2)/2$ [HM09b, Theorem 4.3], so that $d_\ell^* \in [\ln(2), \pi^2 \ln(2)/2] \approx [0.69, 3.42]$. Later the paper [AT09] claimed to have solved the exact value of d_ℓ^*, by stating the following Theorem. However, no proof was given, and no extended version of the paper including a proof has appeared to date.

Theorem 5.5. *[AT09, Theorem 6.3] Let d_ℓ^* be as described above. Then $d_\ell^* = 2\ln(2) \approx 1.39$.*

All these bounds are based on an information-theoretic analysis: if a codelength of less than $2\ln(2)c^2 \ln(1/\epsilon_1)$ is used in *any* probabilistic scheme, then the distributor will not get enough information from the coalition to have enough certainty about guilt of the pirates. This holds regardless of the scheme or code used. The fact that $d_\ell^* = 2\ln(2)$ however only tells us that with unlimited resources (time, calculation power) there should exist a secure scheme with such a codelength. So these bounds may not be very practical, as in practice the time and calculation power available are limited. But knowing this constant does give us ultimate targets, and it tells us how far schemes are from the absolute minimum codelength.

5.2.4 Non-binary alphabets

Very recently, the work in the CREST project at the Eindhoven University of Technology done by Dion Boesten and Boris Skoric has resulted in a paper about the capacity of non-binary fingerprinting channels. This can be compared to the work on d_ℓ^* by Huang and Moulin, but for q-ary alphabets with $q > 2$. This was done for the restricted digit model; a choice we did not have to make for the binary alphabet. The following result about $d_{\ell,q}^*$, the optimal constant for q-ary alphabets, is proven in this paper.

Theorem 5.6. *[BS11, Theorem 3] Let $d_{\ell,q}^*$ be as described above for q-ary alphabets. Then $d_{\ell,q}^* = 2\ln(q)/(q-1)$.*

This last result shows that one can really get better results when switching to a bigger alphabet. For example, a symbol from an alphabet of size 8 can be represented by $\log_2(8) = 3$ bits, by

identifying symbols with triples of bits. So one might expect that the codelength needed will decrease by a factor $\log_2(q)$ when going from a binary to a q-ary alphabet, i.e. $d^*_{\ell,q}/d^*_{\ell,2} = 1/\log_2(q)$. However Theorem 5.6 says that $d^*_{\ell,q}/d^*_{\ell,2} = \log_2(q)/(q-1)$ which is really smaller than $1/\log_2(q)$. For example, using a 64-ary alphabet, one does not have $d^*_{\ell,64} = d^*_{\ell,2}/6 \approx 0.23$ but $d^*_{\ell,64} = 6d^*_{\ell,2}/63 \approx 0.13$. This gain can intuitively be explained by the fact that the bigger the alphabet we use, the more important the choice of the model becomes. For example, for the binary alphabet there is no difference between the restricted digit model and the arbitrary digit model. Then pirates are as strong in the restricted digit model as in the arbitrary digit model. For bigger alphabets, these models are no longer the same, and then using the restricted digit model actually restricts the pirates in their abilities.

5.3 The Boneh-Shaw scheme

5.3.1 Introduction

In this section we will investigate the Boneh-Shaw scheme, which was introduced and analyzed in 1998 by Dan Boneh and James Shaw. This scheme is intuitive, but not very efficient and therefore not the most practical choice. However, this is one of the milestones in collusion-resistant fingerprinting schemes. This scheme was the first probabilistic collusion-resistant scheme with a length polynomial in the number of colluders, and between 1998 and 2003 this was also the best probabilistic static fingerprinting scheme known. Furthermore, later we will again run into the cubic Boneh-Shaw scheme, since Tassa also used this scheme in his paper [Tas05].

First of all, let us give a motivational example of how the (cubic) Boneh-Shaw scheme roughly works, and why. Consider the following code $\mathcal{C}_0$, also given in Table 5.1.

$$\mathcal{C}_0 = \{(111;111;111), (000;111;111), (000;000;111), (000;000;000)\} \tag{5.1}$$

	block 1			block 2			block 3		
Alice	1	1	1	1	1	1	1	1	1
Bob	0	0	0	1	1	1	1	1	1
Charlie	0	0	0	0	0	0	1	1	1
Dave	0	0	0	0	0	0	0	0	0

Table 5.1: *The fingerprinting code $\mathcal{C}_0$ for 4 users with length $\ell = 9$.*

Let $\mathcal{C}$ be the code formed by applying a random permutation π to the columns of the code $\mathcal{C}_0$. We will not consider this permutation for the analysis, as after receiving $\vec{y}$ we can simply invert the permutation, but we do use the fact that *users do not know this permutation π*. This permutation π thus belongs to the secret data described in the model of (static) fingerprinting schemes.

Now consider the coalition formed by all users but the second user. Then on the first six positions, the colluders see $(111;111), (000;000), (000;000)$ respectively. Since a random permutation is used, the colluders cannot distinguish the first three positions from the second three positions. In fact, the only user who can distinguish between those positions is user 2, as that is the only user with differences on those positions. The scheme is based on this observation: if user 2 is innocent, then in the forgery $\vec{y}$ the first three symbols will be similar to the second three symbols. And similarly, if in the forgery there is an odd balance in the number of ones for the first three and second three positions, then most likely user 2 was included in the strategy to form the codeword.

Suppose for example that the colluders use the majority strategy. Then the output will be $\vec{y} = (000; 000; 111)$. As the number of ones on the first three positions is the same as on the second three positions, user 2 is most likely innocent. However, there is a clear difference between the second three and the last three positions, which suggests user 3 is guilty.

Consider instead the scapegoat strategy, where $\vec{y} = \vec{x}_j$ for some member j of the coalition. If $j = 1$, then from the ones on the first three positions, one can deduce that user 1 must be part of the coalition. If $j = 3$, then we get the same as with the majority strategy, and user 3 is likely guilty. If $j = 4$, then since $\vec{y}_7 = \vec{y}_8 = \vec{y}_9 = 0$ one knows that the last user is guilty.

A general reasoning why someone will get suspected goes as follows. On the first three positions, only user 1 sees ones, so unless there are only zeroes, we already have a suspect. Now if these are indeed all zeroes, then we also expect the second three symbols to be zeroes; if there is a significant difference between these two adjacent groups of three symbols, then we know user 2 is most likely part of the coalition. So either user 2 is suspected, or these three symbols also contain many zeroes. This can be continued for the third three positions, so that either these last three symbols are also mostly zeroes, or user 3 is a suspect. But now, since user 4 is the only one with zeroes on these last three symbols, this would imply that this last user is suspicious. So since at the start we expect only zeroes, at the end we expect only ones and inbetween we don't expect big changes in the numer of zeroes and ones between adjacent groups of symbols, one of our expectations will not come true, and someone will be suspected. In other words, the number of ones seen in each block should start at 0, end at 3 and should not grow too fast inbetween. With sufficiently large block sizes this then leads to a contradiction, which means someone will get accused.

5.3.2 The cubic Boneh-Shaw scheme

Let us now define the scheme properly, and analyze its properties. First of all, we use d to denote the duplication factor, which was 3 in the above example. The above construction then leads to a code, where user j gets the codeword $\vec{x}_j$ with $(\vec{x}_j)_i = 0$ for the first $d(j-1)$ positions and $(\vec{x}_j)_i = 1$ for the remaining $d(n-j)$ positions. Note that this gives a total of $\ell = d(n-1)$ positions. We write B_k for the k^{th} block of d positions. Identifying user j (for $2 \leq j \leq n-1$) then relies on comparing blocks B_j and B_{j+1}, while identifying users 1 and n relies solely on counting the number of ones in the blocks B_1 and B_{n-1} respectively.

We define the accusation algorithm as follows. Let w_i be the weight of $\vec{y}$ on block B_i, i.e. w_i is the number of ones on the positions $d(i-1)+1$ up to di in $\vec{y}$. If $w_1 > 0$ then obviously user 1 is accused and guilty, and if $w_{n-1} < d$ then the last user must be guilty and gets accused. For the other users, let $a_i = (w_{i-1} + w_i)/2$ be the average number of ones in the blocks B_{i-1} and B_i. We now accuse user j if the following condition holds:

$$w_j - w_{j-1} > 2\sqrt{\frac{a_j}{2} \log\left(\frac{2n}{\epsilon_1}\right)}. \tag{5.2}$$

For this scheme, we then get the following result about soundness.

Theorem 5.7. *[BS98, Lemma V.2] The probability of accusing any innocent user in the cubic Boneh-Shaw scheme is at most ϵ_1/n. Hence with probability at least $1 - \epsilon_1$, no innocent users are accused.*

Proof. Suppose j is an innocent user. Obviously if $j = 1$ (or $j = n$), then for the coalition the marking assumption applies on the first (last) d positions, so that $w_1 = 0$ ($w_{n-1} = d$). So users 1 and n are never falsely accused.

Now suppose $2 \leq j \leq n-1$ is an innocent user, and suppose that the forgery $\vec{y}$ contains a ones on the blocks $B_j \cup B_{j-1}$. Since the permutation is uniformly random, the ones in these two blocks are uniformly random spread throughout the two blocks B_j and B_{j-1}. Let Y denote the number of ones in block B_{i-1}, given that there are a ones in the two blocks together. Then the distribution of Y is given as $\mathbb{P}[Y = r] = \binom{d}{r}\binom{d}{a-r}/\binom{2d}{a}$. Obviously Y has mean $a/2$.

Let X denote a binomial random variable over d experiments with success probability $1/2$. Then $\mathbb{P}[Y = r] \leq 2\mathbb{P}[X = r]$ for any r. Hence for any r we get $\mathbb{P}[d/2 - Y > r] \leq \mathbb{P}[d/2 - X > r] \leq 2e^{-2r^2/d}$, where the last inequality comes from the Chernoff bound for binomial random variables with success probability $1/2$. Taking $r = \sqrt{a_j \log(2n/\epsilon_1)/2}$ the bound thus gives us:

$$\mathbb{P}\left[d/2 - Y > \sqrt{\frac{a_j}{2}\log\left(\frac{2n}{\epsilon_1}\right)}\right] \leq 2e^{-\log(2n/\epsilon_1)} = \epsilon_1/n. \tag{5.3}$$

So user j is declared guilty with probability at most ϵ_1/n. Therefore the probability that no innocent user is declared guilty is at least $(1-\epsilon_1/n)^n \geq 1-\epsilon_1$, which proves the result. □

While the above proves soundness, we also have to consider the completeness property. As mentioned in the Introduction of this chapter, the code is built in such a way that there have to be two adjacent blocks somewhere which are really different, i.e. the numbers of ones in both blocks are far apart. We will prove that if d is sufficiently large, the set of accused users is not empty, as was also proven in [BS98, Lemma V.3]. This then implies that with probability at least $1-\epsilon_1$, the set of accused users is not empty and includes no innocent users. Therefore the set of accused users must contain at least one guilty user, so the completeness property then holds with error probability at most $\epsilon_2 = \epsilon_1$.

Theorem 5.8. *[BS98, Theorem V.1] Using $d = 2n^2\log(2n/\epsilon_1)$, the set of accused users is always non-empty. Hence the probability of not accusing any guilty users is at most $\epsilon_2 = \epsilon_1$.*

Proof. Suppose no one is accused. First we prove by induction that for all i it must then hold that $w_i \leq 2i^2\log(2n/\epsilon_1)$. This then implies that $w_{n-1} \leq 2(n-1)^2\log(2n/\epsilon_1)$, while $w_{n-1} = 2n^2\log(2n/\epsilon_1)$, since user $n-1$ is not accused. This then gives a contradiction and completes the proof. So it only remains to show that $w_i \leq 2i^2\log(2n/\epsilon_1)$ for all i.

First of all, since user 1 is not accused we know that $w_1 = 0 \leq 2\cdot 0^2\log(2n/\epsilon_1)$, as required. Now suppose $w_i \leq 2i^2\log(2n/\epsilon_1)$. Then $w_{i+1} = a_{i+1} - w_i \leq a_{i+1}/2 + \sqrt{a_{i+1}\log(2n/\epsilon_1)/2}$, since user i was not accused. Substituting $a_{i+1} = w_i + w_{i+1}$ for both occasions of a_{i+1}, and again using the bound on w_i, we can isolate w_{i+1} as $w_{i+1} \leq 2i^2\log(2n/\epsilon_1) + 2\sqrt{2(2i^2\log(2n/\epsilon_1) + w_{i+1})\log(2n/\epsilon_1)}$. If we now suppose that $w_{i+1} = 2r^2\log(2n/\epsilon_1)$ for some r, then $r^2 \leq i^2 + \sqrt{r^2+i^2}$. From this it follows that $r \leq i+1$, so that $w_{i+1} \leq 2(i+1)^2\log(2n/\epsilon_1)$ as was to be proven. □

The scheme described above thus gives a code of length $d(n-1) = 2n^2(n-1)\log(2n/\epsilon_1) = \mathcal{O}(n^3\log(n/\epsilon_1))$ and an efficient accusation algorithm, which satisfies the soundness and completeness properties with error probabilities bounded by $\epsilon_1 = \epsilon_2$.

5.3.3 The quartic Boneh-Shaw scheme

In the previous subsection we discussed a cubic scheme with length $\ell = \mathcal{O}(n^3\log(n/\epsilon_1))$. However, this codelength is cubic in the number of users, rather than the number of pirates. Especially with large systems with many users and only few colluders, almost any scheme with a length polynomial in c instead of n would be better than the above scheme. Here we will discuss

an extension of the cubic scheme into a quartic scheme, but with the n replaced by a c. The codelength then becomes $\mathcal{O}(c^4 \log(n/\epsilon_1))$ which is much better than $\mathcal{O}(n^3 \log(n/\epsilon_1))$ for $c \ll n$.

The quartic Boneh-Shaw scheme is constructed as follows. Let $\mathcal{C}_1$ be the cubic Boneh-Shaw as described in the previous subsection, with length ℓ_1 and consisting of n_1 codewords. We now concatenate this code with a uniformly random code $\mathcal{C}_2$ over an alphabet of size n_1. This code has some codelength ℓ_2 and cardinality n. The concatenated code $\mathcal{C}$ thus consists of n codewords of length $\ell_1\ell_2$, where each codeword is made up of ℓ_1 random codewords from $\mathcal{C}_1$.

The accusation algorithm is now slightly different. For each of these ℓ_1 components of the longer codeword, we run the accusation algorithm from the cubic Boneh-Shaw scheme, and we arbitrarily choose one of the accused users from this run as the accused user for this component. We do this for every component, thus getting ℓ_1 accusations of a single user. Putting these ℓ_1 accusations together, we get a word $\sigma_1\sigma_2 \dots \sigma_{\ell_1}$ with each σ_i a number (user) between 1 and n. We now look for the word $\vec{x} \in \mathcal{C}$ which matches this word $\sigma_1 \dots \sigma_{\ell_1}$ on the most positions. If there is a tie, we just arbitrarily choose one. Finally, we accuse the user associated to this codeword $\vec{x}$. For this scheme we then get the following result.

Theorem 5.9 (Quartic Boneh-Shaw scheme)**.** *[BS98, Theorem V.5] Let n, c, ϵ_1 be given. The quartic Boneh-Shaw scheme with parameters $n_1 = 2c, \ell_1 = 2c\log(2n/\epsilon_1)$ and $d = 2n_1^2 \log(4n_1\ell_1/\epsilon_1) = 8c^2 \log(8c\ell_1/\epsilon_1)$ accuses exactly one user, who is guilty with probability at least $1 - \epsilon_1$, provided that the forgery was generated by a coalition of at most c users. This scheme has a codelength of:*

$$\ell = \ell_1 d(n_1 - 1) = 32c^4 \left(1 - \frac{1}{2c}\right) \log\left(\frac{2n}{\epsilon_1}\right) \log\left(\frac{16c^2}{\epsilon_1} \log\left(\frac{2n}{\epsilon_1}\right)\right) \tag{5.4}$$

$$= \mathcal{O}\left(c^4 \log\left(\frac{n}{\epsilon_1}\right) \log\left(\frac{c}{\epsilon_1}\right)\right). \tag{5.5}$$

Sketch of the proof. Using the analysis for the cubic Boneh-Shaw scheme, we know that for every segment one of the coalition members will be accused with probability at least $1 - \epsilon_1/2\ell_1$, as the factor inside the logarithm of d is now $2\ell_1$ times bigger compared to the cubic Boneh-Shaw scheme. So the probability that on *all* positions a coalition member will be accused is at least $1 - \epsilon_1/2$.

Now since there are c users and ℓ_1 segments, we know that if indeed all segments are translated to coalition members, then at least one of the coalition members must match the output word $\vec{\sigma} = \sigma_1 \dots \sigma_{\ell_1}$ on at least ℓ_1/c positions. Any random word, belonging to an innocent user, is expected to match $\vec{\sigma}$ on $\ell_1/n_1 = \ell_1/2c$ positions. The probability that such a word does match $\vec{\sigma}$ on more positions than all members of the coalition, thus on at least ℓ_1/c positions, can again be bounded using the Chernoff bound. This probability is then at most $\epsilon_1/2n$. Therefore the probability that all innocent users match $\vec{\sigma}$ on less than ℓ_1/c positions is at least $1 - \epsilon_1/2$. So with probability at least $(1 - \epsilon_1/2)(1 - \epsilon_1/2) \geq 1 - \epsilon_1$, all segments are correctly translated to coalition members, and no innocent users have as many as ℓ_1/c matches with $\vec{\sigma}$. So with probability at least $1 - \epsilon_1$, the algorithm outputs a coalition member. □

Note that the above scheme achieves only $\ell = \mathcal{O}(c^4 \log(n/\epsilon_1) \log(c/\epsilon_1))$, while the lower bound says $\ell = \Omega(c^2 \log(1/\epsilon_1))$. So the above scheme has a length which is roughly the square of the length of the theoretical lower bound. Also, the constant in front of c^4 is 32, so even for, say, $c = 5$, the length is already at least 20000, where we did not even take into account the logarithmic factors. With $n = 10^6$ and $\epsilon_1 = 10^{-3}$ for example, $\log(n/\epsilon_1) \approx 30$ and $\log(1/\epsilon_1) \approx 10$, so that the codelength is at least six million. Compared to $2\ln(2)c^2 \log(n/\epsilon_1) < 1000$, this is of course terrible.

5.3.4 Limitations

One way to get better schemes would be to retrace our steps in the Boneh-Shaw scheme, and see if we can improve the scheme in some way, e.g. by sharpening the analysis or making better parameter choices. This was done in several papers, e.g. [Yac01] and [Sch06]. However, the scheme's c^4 is too big, and in 2003 the paper [PSS03] actually proved that there are limitations to the Boneh-Shaw scheme. One can shave off at most a factor c, as the following Theorem states.

Theorem 5.10. *[PSS03, Theorem 4.1] Let $\mathcal{C}$ be a Boneh-Shaw code, and let σ be the associated Boneh-Shaw accusation algorithm. If the scheme $(\mathcal{C}, \sigma)$ is secure against c colluders with at most $\epsilon_1 = \epsilon_2$ error, then $\ell = \Omega(c^3 \log(n/c\epsilon_1))$.*

Even with modifications, such as using other types of columns, we will not get close to the bound of $\Omega(c^2 \log(n/\epsilon_1))$, as the following Theorem states. If we only use a limited number of different columns for our code, and roughly follow the Boneh-Shaw scheme, then we will always need a codelength of roughly $\Omega(c^3 \log(n/\epsilon_1))$. Note that even for $k = \ell$, i.e. all columns different, the result is still in the order of c^3. This means that we need a different way of accusing users, and possibly a completely different scheme altogether, if we want to get shorter codelengths.

Theorem 5.11. *[PSS03, Theorem 6.1] Let $\mathcal{C}$ be a Boneh-Shaw-like multiplicity code with k types of columns, each repeated d times. Let σ be some accusation algorithm, such that the security of the scheme depends entirely on the secrecy of the permutation π applied to the columns of the code, and not on the secrecy of the columns themselves. Then if $(\mathcal{C}, \sigma)$ is secure against c colluders with at most $\epsilon_1 = \epsilon_2$ error, then $\ell = \Omega(c^3 \log(n/c\epsilon_1)/\log(k))$. In particular, as $k \leq \ell$ is at most polynomial in c, we always have $\ell = \Omega(c^3 \log(n/c\epsilon_1)/\log(c))$.*

The above shows us that there are serious limitations to the Boneh-Shaw scheme. To get close to the lower bounds, we will need to use a completely different approach.

5.3.5 Summary

The Boneh-Shaw scheme is a relatively simple and intuitive scheme, and it is the first probabilistic static scheme to get a codelength polynomial in c and $\log(1/\epsilon_1)$. The cubic scheme achieves a codelength $\mathcal{O}(c^3 \log(n/\epsilon_1))$, provided that $n = \mathcal{O}(c)$. If $n \gg c$ we can use the quartic scheme to get a codelength of $\ell = \mathcal{O}(c^4 \log(n/\epsilon_1) \log(c/\epsilon_1))$. However, the factor c^4 grows quadratically as fast as the lower bound of c^2. Even for small c, for instance $c = 5$, the factor c^4 is already 25 times bigger than c^2. Also taking into account the large constant 32 in front of the c^4, this gives a length of more than 500 times more than the sharpest lower bound for $c = 5$, and even larger factors for larger c.

While one can look for improvements to reduce the codelength, the results by Peikert et al. in [PSS03] are most interesting. These results show that there are limitations to the scheme in general, and that no improvement can get the codelength below $\ell = \Omega(c^3 \ln(n/\epsilon_1))$. Therefore we did not spend time investigating such improvements, as it will not get us the results we want anyway, i.e. a codelength quadratic in the number of colluders.

Concluding, we can say that investigating the scheme was a useful exercise, but we also learned that we have to do something different if we want to get closer to the theoretical optimal codelength.

5.4 The Tardos scheme

5.4.1 Introduction

In this section we will discuss the Tardos scheme, which was proposed in 2003 by Gabor Tardos. As this scheme is very important, both theoretically and practically, we will spend quite some time on explaining the ideas behing this scheme. Also, this scheme is the basis for all four chapters of Part II, so of all schemes discussed in this literature, this is by far the most important one for us.

First of all, In this Introduction we will consider a sketch of the Tardos fingerprinting scheme to get a feeling for how the scheme works and why it should work. In this sketch we will leave out most of the details, to emphasize the bigger picture. For this, we first look at a sketch of the construction, which roughly explains how the code is generated and how the accusation algorithm works. Then we will discuss why this scheme should intuitively work. Finally we look at an "example", with certain chosen (small) parameters, to illustrate the scheme and to get an idea of how the scheme works in practice.

In the next subsection we will then look at the original Tardos scheme, presented in [Tar03]. Then in Subsection 5.4.3 we will look at the improvements suggested in various papers to make the Tardos scheme even better. Then finally we will wrap up with a summary of the results from this section.

5.4.1.1 Construction

So first let us look at a sketch of the construction. We assume c and n are known, and we assume we have been given some upper bounds ϵ_1, ϵ_2 on the false positive/negative probabilities respectively.

For the initialization, we first take ℓ sufficiently large, and we choose a parameter $\delta > 0$ sufficiently small and a parameter Z between certain upper and lower bounds. The value of δ is close to zero, while Z grows linearly in c and ℓ grows quadratically in c.

Then for every fingerprint position $1 \leq i \leq \ell$, we pick a *bias* p_i from $[\delta, 1-\delta]$ from some distribution function $F = F_c$. All these biases are selected independently. The distribution F is biased towards δ and $1-\delta$, e.g. the probability of getting $p_1 \in [\delta, \delta + 0.02]$ is bigger than getting $p_1 \in [0.49, 0.51]$. The associated probability density function f is also symmetric around $1/2$, i.e. $f = f_c$ satisfies $f(1/2 - a) = f(1/2 + a)$ for all a.

Now comes the codeword generation. For this, we select the i^{th} symbol of user j randomly from a Bernoulli distribution with bias p_i. In other words, with probability p_i we take $X_{ji} = 1$, and with probability $1 - p_i$ we take $X_{ji} = 0$. Note that X_{ji} and X_{ki} are independent for $j \neq k$, and that also X_{ji} and X_{jk} are independent for all $i \neq k$. This means that the codewords of different users are independent, and that the different codeword positions are independent. Especially the former is an important and very useful property of this code, since it basically rules out any attempts of framing users.

After the above steps the initialization and codeword generation is completed. Table 5.2 summarizes this process graphically. First the values p_i are generated, and then the symbols X_{ji} are chosen, resulting in a code $\mathcal{C}$.

After sending out the codewords and receiving some forgery $\vec{y}$, the distributor calculates scores S_j for all users. This S_j can be seen as a sum of accusation scores S_{ji} over all positions i, i.e. $S_j = \sum_i S_{ji}$, where the values of S_{ji} depend solely on p_i, the pirate output y_i and the user symbol $(\vec{x}_j)_i = X_{ji}$. The value S_j measures the user's suspicion in our system; if S_j is large,

	$p_1 \sim F$ (position 1)	$p_2 \sim F$ (position 2)	... ...	$p_\ell \sim F$ (position ℓ)
Alice	$X_{1,1} \sim \text{Ber}(p_1)$	$X_{1,2} \sim \text{Ber}(p_2)$	...	$X_{1,\ell} \sim \text{Ber}(p_\ell)$
Bob	$X_{2,1} \sim \text{Ber}(p_1)$	$X_{2,2} \sim \text{Ber}(p_2)$	...	$X_{2,\ell} \sim \text{Ber}(p_\ell)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Zoey	$X_{n,1} \sim \text{Ber}(p_1)$	$X_{n,2} \sim \text{Ber}(p_2)$	...	$X_{n,\ell} \sim \text{Ber}(p_\ell)$

Table 5.2: *The construction of the Tardos fingerprinting code. For each column i, the distributor first selects a value p_i from the distribution F. Then for each user he generates a symbol (either a 0 or a 1) by using p_i as the probability of taking a 1. Hence each user's codeword is independent, all columns are independent, and in row i we will roughly see $n \cdot p_i$ ones and $n \cdot (1 - p_i)$ zeroes. The code X and the values p_i are kept secret from the users.*

then he is suspicious, while if S_j is negative or small, we do not suspect user j. If $p_i = \delta$ is small and $y_i = 1$ (which is unlikely), then the value S_{ji} is a big positive number if $(\vec{x}_j)_i = 1$ and S_{ji} is a small negative number if $(\vec{x}_j)_i \neq 1$. If on the other hand $y_i = 0$ (which is much more likely, given p_i is small), then S_{ji} is a big negative number if $(\vec{x}_j)_i = 0$ and S_{ji} is a small positive number if $(\vec{x}_j)_i \neq 1$. Table 5.3 below summarizes these different contributions to S_j for different values of p_i, X_{ji} and y_i.[2]

As mentioned, this score S_{ji} is independent of any information from other users. The total score S_j only depends on the values of p_i, the vector $\vec{y}$ and the vector $\vec{x}_j$. This is in particular useful for proving the soundness property, as $\vec{y}$ and $\vec{x}_j$ are independent if user j was not part of the coalition.

After calculating these scores, the distributor simply accuses all those users whose scores are too high, namely all users j with $S_j > Z$. Therefore Z is also known as the *accusation threshold* or *offset*.

5.4.1.2 Discussion

Now that we have looked at an outline of the scheme and understand a bit how the scheme works, some questions arise, such as: Why does the scheme above make sense? And in particular, two related questions are important: Why are no innocent users accused? and Why will at least one guilty user be accused? We will intuitively discuss these two questions below, disregarding most of the mathematical computations done later to prove the security of this scheme.

For the first question, we use a fact that we have not mentioned before, namely that $\mathbb{E}[S_{ji}] = 0$ and $\text{Var}[S_{ji}] = 1$, provided that X_{ji} and y_i are independent. So then for innocent users j we have $\mathbb{E}[S_j] = \mathbb{E}[\sum_{i=1}^{\ell} S_{ji}] = \sum_{i=1}^{\ell} \mathbb{E}[S_{ji}] = 0$. For the variance we use that p_i and $p_{i'}$ are independent for $i \neq i'$, so that $\text{Var}[S_j] = \text{Var}[\sum_{i=1}^{\ell} S_{ji}] = \sum_{i=1}^{\ell} \text{Var}[S_{ji}] = \ell$, hence the standard deviation is given by $\sigma = \sqrt{\ell}$. As we will see later Tardos chooses $Z > 2\sqrt{\ell}$, and one can imagine that most of the data of some distribution will be at most two standard deviations from the mean. So the probability that $S_j > Z$, i.e. an innocent user j is accused, will be reasonably small.

For the second question, note that by taking the distribution for the p_i's such that $f(p)$ is large for p_i close to 0 and 1, we ensured that quite often all pirates receive the same symbol and the marking assumption applies. If this is the case, then the scores of all pirates will increase, so the total coalition score $S = \sum_{j \in C} S_j$ will increase a lot. In other cases, where pirates are allowed

[2] In the original Tardos scheme, the score S_{ji} was taken as 0 for the top half of the table. Intuitively the scores S_{ji} should be as in the table, making the accusation scores symbol-symmetric. This is one of the improvements, discussed in 5.4.3.

	$p_i \approx \delta$ (low rate of ones)	$p_i \approx 1/2$ (medium rate of ones)	$p_i \approx 1 - \delta$ (high rate of ones)
$X_{ji} = 0$ $y_i = 0$	$S_{ji} \approx +\sqrt{\delta} \approx +0$ Most users have a 0 on this position, including the user and the out of the pirates. This could very well be a coincidence, so the user is only slightly suspicious, and S_{ji} contributes a small positive amount to the sum S_j.	$S_{ji} \approx +1$ Some users have the symbol 0 here and some have a 1 here. The fact that user j and the forged copy match on this position makes the user somewhat suspicious. Therefore S_{ji} is a positive number.	$S_{ji} \approx +\sqrt{1/\delta} \approx +20\sqrt{c}$ Most users have a 1 here, but user j and the pirated copy have a 0 here. This is a very unlikely coincidence, so this definetely makes the user suspicious. So S_{ji} is a big positive number, making the user a strong suspect.
$X_{ji} = 1$ $y_i = 0$	$S_{ji} \approx -\sqrt{1/\delta} \approx -20\sqrt{c}$ Most users have a 0 on this position, including the pirates, but user j has a 1 here. This certainly pleads for his innocence, so S_{ji} is a big contribution to proving the user's innocence.	$S_{ji} \approx -1$ Some users have a 0 here and some users have a 1 on this position. The fact that user j has a 1 here while the pirated copy has a 0 here pleads for the user, so S_{ji} is a negative number.	$S_{ji} \approx -\sqrt{\delta} \approx -0$ Most users have a 1 here, including user j, while the pirated copy has a 0 on this position. This makes the user a bit less suspicious, so S_{ji} is a small contribution to proving his innocence.
$X_{ji} = 0$ $y_i = 1$	$S_{ji} \approx -\sqrt{\delta} \approx -0$ Most users have a 0 here, including user j, while the pirated copy has a 1 on this position. This makes the user a bit less suspicious, but this could also be a coincidence. So S_{ji} is only a small negative contribution to proving his innocence.	$U_{ji} \approx -1$ Some users have a 0 here and some users have a 1 on this position. The fact that user j has a 0 here while the pirated copy has a 1 here pleads for the user, so S_{ji} is a negative number.	$S_{ji} \approx -\sqrt{1/\delta} \approx -20\sqrt{c}$ Most users have a 1 on this position, including the pirates, but user j has a 0 here. This certainly pleads for his innocence, so S_{ji} is a big negative number, i.e. a big contribution to proving the user's innocence.
$X_{ji} = 1$ $y_i = 1$	$S_{ji} \approx +\sqrt{1/\delta} \approx +20\sqrt{c}$ Most users have a 0 here, but user j and the pirated copy have a 1 here. This is a very unlikely coincidence, so this definetely makes the user suspicious. So S_{ji} is a big positive number, making the user a strong suspect.	$S_{ji} \approx +1$ Some users have a 0 and some have a 1 here. The fact that user j and the pirated copy match on this position makes the user somewhat suspicious. Therefore S_{ji} is a positive number.	$S_{ji} \approx +\sqrt{\delta} \approx +0$ Most users have a 1 on this position, including user j and the pirates. This could very well be a coincidence, so the user is only slightly suspicious, and S_{ji} only contributes a small positive amount to the sum S_j.

Table 5.3: *The increase in the score of user j on position i, for several values of X_{ji}, y_i and p_i. If $X_{ji} = y_i$ then S_{ji} will be positive, while for $X_{ji} \neq y_i$ the value of S_{ji} will be negative. Furthermore the contribution of S_{ji} to the total sum is determined by how rare the occurring event is, where rare events cause larger contributions than commonly occurring events.*

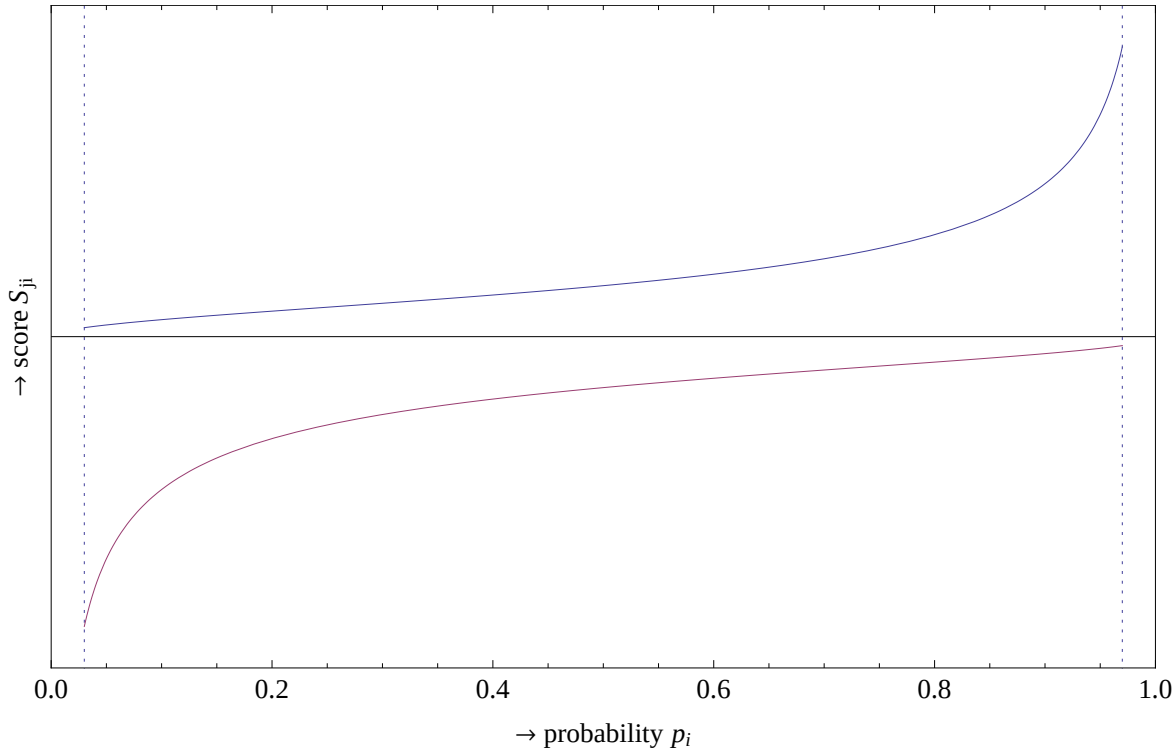

Figure 5.1: *The score S_{ji} for $y_i = 0$ as a function of p_i, for $X_{ji} = 0$ (top) and for $X_{ji} = 1$ (bottom). Because of the cutoff parameter δ, the value of S_{ji} is bounded from above and below by $-\sqrt{(1-\delta)/\delta} \leq S_{ji} \leq +\sqrt{(1-\delta)/\delta}$, where $\sqrt{(1-\delta)/\delta} = \mathcal{O}(\sqrt{c})$.*

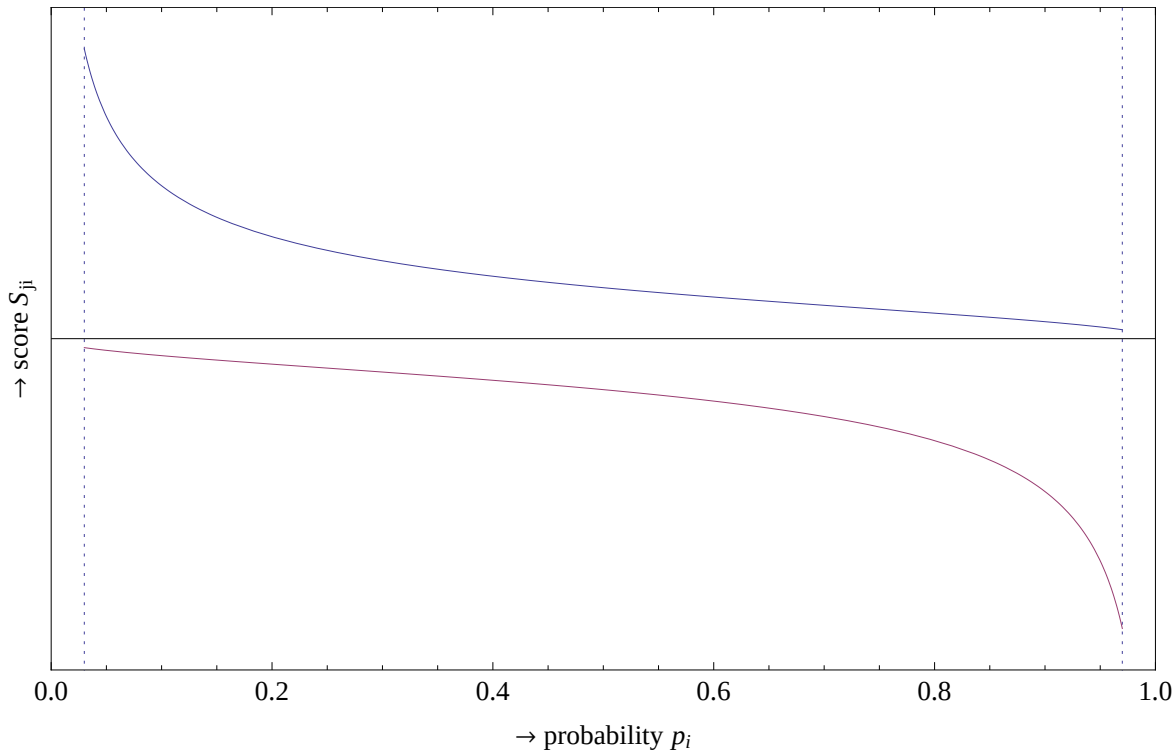

Figure 5.2: *The score S_{ji} for $y_i = 1$ as a function of p_i, for $X_{ji} = 1$ (top) and for $X_{ji} = 0$ (bottom). Because of the relation $S_{ji}(y_i, X_{ji}, p_i) = S_{ji}(1 - y_i, 1 - X_{ji}, 1 - p_i)$, the graph looks very similar to the graph for the case $y_i = 0$.*

to choose their output, the sum S may decrease slightly. However, the decrease in S on those positions is really small in most cases, regardless of whether the pirates output a 0 or a 1. In fact, if the number of ones seen on some position i is exactly p_i, then both $y_i = 0$ and $y_i = 1$ lead to no change in S. Only if e.g. they receive many ones while p_i is small, and they output a 0 will their score decrease a lot. But that happens only rarely, and it doesn't compensate for the increase in S on the positions where the marking assumption applies.

In other words, the reason why pirates will get accused in this scheme is that (a) the total score S of all users added together increases a lot on undetectable positions where the marking assumption applies, and (b) there is no way for pirates to make up for this by decreasing S on detectable positions. Eventually S will get arbitrarily large, so that at some point $S > cZ$. This then implies that for at least one coalition member j his score S_j must be larger than Z, so that at least one of the pirates is accused.

We will see in the next subsection that we can actually prove that, no matter which strategy the pirates choose, the probability of accusing at least one of the guilty users is always very high and that the probability of accusing innocent users is sufficiently small. For now, we will first continue with a rough example of how the whole Tardos scheme works for some small parameter choices.

5.4.1.3 Example

Let us illustrate the scheme through a very small example, where we simply choose some small parameters. Let the system contain $n = 5$ users, and let us use a codelength of $\ell = 6$. We choose the accusation offset as $Z = 1$, and randomly choose the biases as $\vec{p} = (p_1, \ldots, p_6) = \frac{1}{10}(8, 7, 2, 1, 5, 3)$.[3] Suppose the matrix X was randomly selected as:

$$X = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.6}$$

This corresponds to the following distribution of codewords.

	$p_1 = 0.8$	$p_2 = 0.7$	$p_3 = 0.2$	$p_4 = 0.1$	$p_5 = 0.5$	$p_6 = 0.3$
Alice	1	1	0	0	0	1
Bob	0	1	0	0	1	0
Charlie	1	0	0	0	1	1
Dave	1	0	1	0	0	0
Eve	1	1	0	0	1	0

Table 5.4: *The Tardos fingerprinting code for* 5 *users and* $\ell = 6$*, as described by* X *and* $\vec{p}$ *above.*

Then the initialization and codeword generation is complete. Let us now consider some values for the forgery $\vec{y}$, and see which users are accused in those cases.[4]

[3] These biases have to be taken according to F, which depends on t, which in turn depends on c, which has not even been specified. For the sake of simplicity we will ignore these issues and just assume that these p_i's take on the given nice round values and that $\ell = 6$.

[4] For the accusation score function we will use the accusation function suggested in [SKC08], which is the most intuitive and has the best performance.

- Let $\vec{y} = (0, 1, 1, 0, 0, 1)$. Then $\vec{S} = (S_1, \ldots, S_5) \approx \frac{1}{10}(25, 8, -17, 7, -17)$ so only user 1 is accused. Note that $\vec{x}_1$ and $\vec{y}$ match on four out of the six positions. Similarly, the users with the lowest scores (users 3 and 5) have the least matches with $\vec{y}$, namely two.

- Let $\vec{y} = (1, 0, 0, 1, 1, 0)$. Then $\vec{S} \approx \frac{1}{10}(-25, -8, 17, -7, 17)$ so users 3 and 5 are accused. Note that the scores are the same as for $\vec{y} = (0, 1, 1, 0, 0, 1)$, except for the multiplication with -1. Also note that users 3 and 5 could not even have generated this codeword together. And since there is a 1 on the fourth position and none of the users have a 1 there, no coalition could ever have generated this codeword under our assumptions. Still, the 'probability' that users 3 and 5 are guilty is high enough for the scheme to accuse them.

- Let $\vec{y} = (0, 0, 0, 0, 0, 0)$. Then $\vec{S} \approx \frac{1}{10}(-8, 18, -7, 10, -7)$ and users 2 and 4 are accused. In the original Tardos scheme with Tardos' original accusation function, we would have $\vec{S} = \vec{0}$ always if $\vec{y} = \vec{0}$. So in the original scheme, pirates would get away if they were able to generate the all-zero codeword. But the probability of being able to generate this word is $\prod_{i=1}^{\ell}(1 - p_i^c)$, which quickly converges to zero as quite many values p_i are close to 0.

- Let $\vec{y} = (1, 1, 1, 0, 0, 0)$. Then $\vec{S} \approx \frac{1}{10}(5, -19, -37, 30, 6)$ and user 4 is accused.

- Let $\vec{y} = (1, 1, 0, 0, 0, 0)$. Then $\vec{S} \approx \frac{1}{10}(15, -9, -27, -10, 16)$ and users 1 and 5 are accused. Notice the big changes in the scores compared to the previous $\vec{y}$. For instance, the score of user 4 went from the highest score by far to the second lowest score.

- Let $\vec{y} = (0, 0, 1, 0, 0, 0)$. Then $\vec{S} \approx \frac{1}{10}(-18, 8, -17, 50, -17)$ and user 4 is accused. The score of user 4 is extremely high, which can be explained by the fact that $\vec{y}$ and $\vec{x}_4$ match on five of the six positions.

Of course, in reality ℓ is much bigger than 6 and Z is much larger as well, so that the code achieves given maximum error rates. For example, in the original Tardos scheme, if $c = 2$ and $\epsilon_1 = \epsilon_2 = e^{-1} \approx 0.37$, then already $\ell \geq 400$, regardless of n. So even though the Tardos scheme achieves a good asymptotic code length, this does not mean that the value of ℓ is "small". The codelength is still quite large, even for small values of c and large allowed error probabilities ϵ_1 and ϵ_2.

5.4.1.4 Summary

We hope that through this Introduction, the reader could acquaint himself with the Tardos scheme, and that the reader now has a feeling for how the scheme works and why it works. In the next subsection we will continue with the original Tardos scheme given in 2003, and analyzing in detail why the scheme works. This analysis will be quite lengthy, but we hope that by first reading this subsection, the reader is able to see the bigger picture and will be able to follow the analysis from the next subsections more easily.

5.4.2 The original Tardos scheme

In this subsection we will discuss the Tardos fingerprinting scheme given in [Tar03], and prove that Tardos' fingerprinting code is c-secure with (ϵ_1, ϵ_2)-error for his choice of parameters. We first give the full construction, with the parameters given explicitly. Then we will prove that the probability of accusing one or more innocent users is bounded by ϵ_1. Finally we will prove that the probability of not accusing any guilty users is at most ϵ_2.

5.4.2.1 Construction

The complete construction, with the parameters chosen according to the original Tardos scheme, is as follows. Let $n, c \geq 2$ be given integers, and let $\epsilon_1 \in (0,1)$ be the desired upper bound for the error rate. Let us also write $k = \lceil \log(n/\epsilon_1) \rceil$. Then the Tardos fingerprinting scheme works as follows.

1. **Initialization**

   (a) Take $\ell = 100c^2k$ as the code length, and take the parameters δ and Z as $\delta = 1/(300c)$ and $Z = 20ck$. Compute $\delta' = \arcsin(\sqrt{\delta})$ such that $0 < \delta' < \pi/4$.

   (b) For each fingerprint position $1 \leq i \leq \ell$, choose p_i independently from $[\delta, 1-\delta]$ according to the distribution given by the following cumulative distribution function F:

   $$F(p) = \frac{2\arcsin(\sqrt{p}) - 2\delta'}{\pi - 4\delta'}. \tag{5.7}$$

   The probability density function f of this distribution is given by:

   $$f(p) = \frac{1}{(\pi - 4\delta')\sqrt{p(1-p)}}. \tag{5.8}$$

   This function is biased towards δ and $1-\delta$ and symmetric around $1/2$.

2. **Codeword generation**

   (a) For each position $1 \leq i \leq \ell$ and for each user $1 \leq j \leq n$, select the entry X_{ji} of the code matrix X independently by $X_{ji} \sim \mathrm{Ber}(p_i)$.

3. **Accusation**

   (a) For each position $1 \leq i \leq \ell$ and for each user $1 \leq j \leq n$, calculate the score S_{ji} according to:

   $$S_{ji} = S_{ji}(y_i, (\vec{x}_j)_i, p_i) = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1,\ y_i = 1; \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0,\ y_i = 1; \\ 0 & \text{otherwise.} \end{cases} \tag{5.9}$$

   (b) For each user $1 \leq j \leq n$, calculate the total accusation sum $S_j = \sum_{i=1}^{\ell} S_{ji}$. User j is accused if and only if $S_j > Z$.

5.4.2.2 Results

With this construction, we can prove the following results.

Theorem 5.12 (Soundness)**.** *Let $j \in U$ be an arbitrary (innocent) user. Let $C \subseteq U \setminus \{j\}$ be a coalition of any size not containing j, and let ρ be any strategy employed by this coalition. Then*

$$\mathbb{P}[j \in \sigma(\rho(X))] < \epsilon_1/n. \tag{5.10}$$

Therefore the probability of not accusing any innocent users is at least $1 - \epsilon_1$, so the scheme is ϵ_1-sound for any c.

Theorem 5.13 (Completeness)**.** *Let $C \subseteq U$ be a coalition of size at most c, and let ρ be any pirate strategy used by the coalition C. Then*

$$\mathbb{P}[C \cap \sigma(\rho(X)) = \emptyset] < (\epsilon_1/n)^{\sqrt{c}/4}. \tag{5.11}$$

Therefore if $(\epsilon_1/n)^{\sqrt{c}/4} \leq \epsilon_2$, the probability of accusing at least one guilty user is at least $1 - \epsilon_2$ and thus the scheme is c-complete with at most ϵ_2 error.

Note that Theorem 5.13 is not sufficient for proving ϵ_2-completeness for any $\epsilon_2 \geq \epsilon_1$, if $\epsilon_1^{\sqrt{c}/4-1} > n^{\sqrt{c}/4}$. This can occur for small values of c and large values of ϵ_1. However for $c \geq 16$ we certainly have $(\epsilon_1/n)^{\sqrt{c}/4} \leq \epsilon_1$, while for $\epsilon_1 = 1/n$ we have $(\epsilon_1/n)^{\sqrt{c}/4} \leq \epsilon_1$ for any $c \geq 4$. In the extended version of Tardos' paper, Tardos proves the following, slightly stronger statement, which proves ϵ_2-completeness for any $c \geq 4$ and any $\epsilon_2 \geq \epsilon_1$.

Theorem 5.14 (Completeness)**.** *Let $C \subseteq U$ be a coalition of size at most c, and let ρ be any pirate strategy used by the coalition C. Then*

$$\mathbb{P}[C \cap \sigma(\rho(X)) = \emptyset] < (\epsilon_1/n)^{c/4}. \tag{5.12}$$

In particular, for any $c \geq 4$ we get $\mathbb{P}[C \cap \sigma(\rho(X)) = \emptyset] < \epsilon_1$, while for $\epsilon_1 \leq 1/n$ we get $\mathbb{P}[C \cap \sigma(\rho(X)) = \emptyset] < \epsilon_1$ for any $c \geq 2$.

The proof of this latter Theorem given in Tardos' extended paper is very similar to the proof of Theorem 5.13, but this proof involves even more bookkeeping than is already the case in the proof of Theorem 5.13. Therefore we will prove the weaker statement and only give a sketch of the proof of Theorem 5.14. We refer the reader to the extended version of [Tar03] for the complete proof of the latter Theorem.

Together Theorems 5.12 and 5.14 imply the following Corollary about the secureness of the original Tardos fingerprinting scheme.

Corollary 5.15 (Secureness)**.** *Let $c \geq 4$, $n \geq c$ and ϵ_1 be given, and take $\epsilon_2 = \epsilon_1$. Then the Tardos fingerprinting scheme given above is $(c, \epsilon_1, \epsilon_1)$-secure and has a length of $\ell = \mathcal{O}(c^2 \log(n/\epsilon_1))$.*

Next we will prove the claimed results. We will mainly follow the proofs given in the paper by Tardos.

5.4.2.3 Soundness

In this subsubsection we will prove Theorem 5.12, which says that no innocent users are accused with probability more than ϵ_1/n. For this proof we will use expectations, probabilities, the Markov inequality and some simple bounds on the function e^x.

Proof of Theorem 5.12. Let $n, c, \epsilon_1, p_i, X_{ji}$ be given according to the Tardos fingerprinting scheme, and let $\vec{y}$ be some forgery generated by some coalition not containing some fixed user j. Then $j \in \sigma(\vec{y})$ if and only if $S_j > Z$, so proving $\mathbb{P}[S_j > Z] < \epsilon_1/n$ proves the theorem.

First, consider $\mathbb{P}[S_j > Z]$. Using Markov's inequality, which says that $\mathbb{P}[X > a] < \mathbb{E}[X]/a$ for positive random variables X, and using the fact that $e^{\alpha x}$ is a strictly increasing positive function for $\alpha > 0$, we get

$$\mathbb{P}[S_j > Z] = \mathbb{P}[e^{\alpha S_j} > e^{\alpha Z}] < e^{-\alpha Z}\mathbb{E}[e^{\alpha S_j}]. \tag{5.13}$$

Next, consider $\mathbb{E}[e^{\alpha S_j}]$. Using some simple rules for the expectation, we can write

$$\mathbb{E}[e^{\alpha S_j}] = \mathbb{E}\left[\prod_{i:y_i=1} e^{\alpha S_{ji}}\right] = \prod_{i:y_i=1} \mathbb{E}\left[e^{\alpha S_{ji}}\right]. \tag{5.14}$$

We now investigate $\mathbb{E}[e^{\alpha S_{ji}}]$, for $\alpha = 1/(10c)$. Since $\alpha S_{ji} \leq \alpha\sqrt{(1-\delta)/\delta} \leq \alpha\sqrt{1/\delta} \leq \sqrt{3} < 1.74$ we can use the bounds $1 + x \leq e^x \leq 1 + x + x^2$ for $x = \alpha S_{ji}$ and $x = \alpha$, since these inequalities hold for all $x < 1.79$. Also using the fact that S_{ji} has expectation 0 and variance 1, we get

$$\mathbb{E}[e^{\alpha S_{ji}}] \leq \mathbb{E}\left[1 + \alpha S_{ji} + \alpha^2 S_{ji}^2\right] = 1 + \alpha\mathbb{E}[S_{ji}] + \alpha^2\mathbb{E}[S_{ji}^2] = 1 + \alpha^2 \leq e^{\alpha^2}. \tag{5.15}$$

So we get an upper bound for $\mathbb{P}[S_j > Z]$ as

$$\mathbb{P}[S_j > Z] < e^{-\alpha Z}\mathbb{E}[e^{\alpha S_j}] = e^{-\alpha Z}\prod_{i:y_i=1} \mathbb{E}\left[e^{\alpha S_{ji}}\right] \leq e^{-\alpha Z}\left(e^{\alpha^2}\right)^{|\{i:y_i=1\}|} \leq e^{-\alpha Z + \alpha^2 \ell}. \tag{5.16}$$

Filling in $\alpha = 1/(10c)$, $Z = 20ck$, $\ell = 100c^2k$ gives that $-\alpha Z + \alpha^2\ell = -k$, so using $k = \lceil n/\epsilon_1 \rceil$ we get

$$\mathbb{P}[S_j > Z] < \epsilon_1/n, \tag{5.17}$$

which was to be proven. The probability that no one gets accused is thus bounded by

$$\mathbb{P}[\sigma(\rho(X)) \subseteq C] = \prod_{j \notin C}(1 - \mathbb{P}[j \in \sigma(\rho(X))]) \geq \prod_{j \notin C}(1 - \epsilon_1/n) \geq (1 - \epsilon_1/n)^n \geq 1 - \epsilon_1. \tag{5.18}$$

This shows that the scheme is ϵ_1-sound for any c. □

Note that we have proven that for any coalition C of any size, and for any strategy used by this coalition, no innocent user is accused with probability more than ϵ_1/n. For proving that the code is (c, ϵ_1)-sound, we only had to prove that for coalitions of *size at most* c, the probability of accusing an innocent user is at most ϵ_1/n. What we have proven is something much stronger, namely that the code is frameproof with at most ϵ_1-error against coalitions of any size. This is a big advantage of using this scheme, as the Tardos scheme is basically fully frameproof.

Also notice that we have not used the definition of F in this proof. In other words: for any distribution function F on $[\delta, 1-\delta]$, Theorem 5.12 holds. Only for the proof of completeness do we need the definition of F.

5.4.2.4 Completeness

Next we discuss why at least one guilty user is accused in the original Tardos fingerprinting scheme. The proof of Theorem 5.13 is given below, and it is much more lengthy than the proof of soundness. It involves a lot of bookkeeping, which does not really help us understand the scheme. For completeness (pun intended) we give the proof here, but unless the reader wants to verify that the author made no mistake he can skip the proof and continue to the more interesting sections.

Note that the proof nowhere uses that the values of p_i are secret. In fact, the proof only upperbounds the scores on detectable positions. So even if a coalition were to know all values of p_i, the scheme would still be secure; as long as the coalition adheres to the marking assumption.

Note also that in this proof there is not much advanced mathematics needed. Again, the Markov inequality is used (now for $e^{-\beta S}$ instead of $e^{+\alpha S}$, with positive β), and then a lot of work is done to bound the resulting expectation value. This again involves working with probabilities and expectations and the bounds for e^x, and it involves some combinatorics, simple integrations and some other simple bounds.

Proof of Theorem 5.13. Without loss of generality, we assume that the coalition consists of the first c users. We also start by introducing some more notation. We write $x_i = x_i(X) = \sum_{j=1}^{c} X_{ji}$ for the number of ones in column i of the code matrix X. Similar to the notation used before, we write $\vec{p} = (p_1, \ldots, p_\ell)$, and we write $q_i = \sqrt{(1-p_i)/p_i}$ and $S = \sum_{j \in C} S_j$. Writing out this last expression gives

$$S = \sum_{j \in C} S_j = \sum_{j=1}^{c} \sum_{i:y_i=1}^{\ell} S_{ji} = \sum_{i=1}^{\ell} y_i(x_i q_i - (n - x_i)/q_i). \tag{5.19}$$

Since for $S > cZ$, it clearly follows that there is at least one j such that $S_j > Z$ and thus user j is accused, we can bound the probability that no guilty user is accused by

$$\mathbb{P}[C \cap \sigma(\rho(X)) = \emptyset] \leq \mathbb{P}[S < cZ] = \mathbb{P}[e^{-\beta S} > e^{-\beta cZ}] < e^{\beta cZ} \mathbb{E}_{\vec{y},X,\vec{p}}[e^{-\beta S}], \tag{5.20}$$

where $\beta > 0$, and where we have also used some techniques which we have already seen in the proof of Theorem 5.12. Now let us consider the expectation value $\mathbb{E}_{\vec{y},X,\vec{p}}[e^{-\beta S}]$. First, we can rewrite this to

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] = \sum_{X=X_0} \mathbb{E}_{\vec{y},\vec{p}}\left[e^{-\beta S}\mathbb{P}[X = X_0]\right], \tag{5.21}$$

where the summation runs over all $\{0,1\}$-matrices X. We can easily calculate $\mathbb{P}[X = X_0]$, since this probability is simply equal to the product of the probabilities that each entry of X is equal to the corresponding entry of X_0 (since all entries are independent). For position (j,i), this probability is p_i if $(X_0)_{ji} = 1$ and $1 - p_i$ if $(X_0)_{ji} = 0$. So writing this out, and writing out S gives

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] = \sum_{X=X_0} \mathbb{E}_{\vec{y},\vec{p}}\left[e^{-\beta S} \prod_{i=1}^{\ell} p_i^{x_i(X_0)}(1-p_i)^{c-x_i(X_0)}\right] \tag{5.22}$$

$$= \sum_{X=X_0} \mathbb{E}_{\vec{y},\vec{p}}\left[\prod_{i=1}^{\ell} p_i^{x_i}(1-p_i)^{c-x_i} \exp\left(-\beta y_i(x_i q_i - (c-x_i)/q_i)\right)\right] \tag{5.23}$$

$$= \sum_{X=X_0} \prod_{i=1}^{\ell} \mathbb{E}_{y_i,p_i}\left[p_i^{x_i}(1-p_i)^{c-x_i} \exp\left(-\beta y_i(x_i q_i - (c-x_i)/q_i)\right)\right]. \tag{5.24}$$

Now since each p_i is identically (and independently) distributed with distribution function F, we can also replace the expectation over p_i by the expectation over p, where p also has distribution F, and write $q = \sqrt{(1-p)/p}$. Then

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] = \sum_{X=X_0} \prod_{i=1}^{\ell} \mathbb{E}_{y_i,p}\left[p^{x_i}(1-p)^{c-x_i} \exp\left(-\beta y_i(x_i q - (c-x_i)/q)\right)\right]. \tag{5.25}$$

Since y_i is either 0 or 1, we introduce $N_{0,x_i} = \mathbb{E}_p\left[p^{x_i}(1-p)^{c-x_i}\right]$ and $N_{1,x_i} = \mathbb{E}_p\left[p^{x_i}(1-p)^{c-x_i} \exp\left(-\beta(x_i q - (c-x_i)/q)\right)\right]$. Since for $x_i = 0$ it follows that y_i must be 0, and for $x_i = c$ it follows that $y_i = 1$, we write $\max^*(N_{0,x_i}, N_{1,x_i})$ to denote N_{0,x_i} if $x_i = 0$, N_{1,x_i} if $x_i = c$ and the maximum of N_{0,x_i} and N_{1,x_i} if $0 < x_i < c$. This is then an upperbound for $\mathbb{E}_p\left[p^{x_i}(1-p)^{c-x_i} \exp\left(-\beta y_i(x_i q - (c-x_i)/q)\right)\right]$. So

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \sum_{X=X_0} \prod_{i=1}^{\ell} \max{}^*(N_{0,x_i}, N_{1,x_i}). \tag{5.26}$$

Since all terms are nonnegative, we can interchange the summation and product, since $\sum_a \prod_b f(a,b) \leq \prod_b \sum_a f(a,b)$ for $f \geq 0$. We can also replace the summation over $\{0,1\}$-matrices X by a summation over values for x_i, the contributions to the summation are the same if the values of x_i are the same. This gives

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \prod_{i=1}^{\ell} \sum_{x_i=0}^{c} \binom{c}{x_i} \max{}^*(N_{0,x_i}, N_{1,x_i}). \tag{5.27}$$

Since the summations over the x_i are all the same, we can replace the x_i by x and replace the product by an exponentiation, so that we get

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} \max{}^*(N_{0,x}, N_{1,x})\right)^{\ell} \tag{5.28}$$

$$\leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x\right)^{\ell}, \tag{5.29}$$

where $M_0 = N_{0,0}$, $M_c = N_{1,c}$, $M_x = \max(N_{0,x}, N_{1,x})$ for $0 < x < c$, and $N_{b,x} = \mathbb{E}_p\left[p^x(1-p)^{n-x}\exp\left(-\beta b(xq-(c-x)/q)\right)\right]$ for $b \in \{0,1\}$ and $0 \leq x \leq n$.

Now we will get an upper bound for the summation, by inspecting M_x. We now choose $\beta = \sqrt{\delta}/c$ so that $-\beta(xq-(c-x)/q) \leq -\beta(0q-(c-0)/q) = \beta c/q \leq \beta c/\sqrt{1-p} \leq \beta c/\sqrt{\delta} < 1.79$, and we can again use the bound $e^u \leq 1+u+u^2$ for $u = -\beta(xq-(c-x)/q)$. Writing $\gamma = xq-(c-x)/q$ this gives

$$\exp\left(-\beta b(xq-(c-x)/q)\right) \leq 1-\beta\gamma+\beta^2\gamma^2. \tag{5.30}$$

Multiplying by $p^x(1-p)^{c-x}$ and taking the expectation on both sides gives

$$N_{1,x} \leq N_{0,x} - \beta N_{2,x} + \beta^2 N_{3,x}, \tag{5.31}$$

where $N_{2,x} = \mathbb{E}_p[p^x(1-p)^{c-x}\gamma]$ and $N_{3,x} = \mathbb{E}_p[p^x(1-p)^{c-x}\gamma^2] \geq 0$.

We will now focus on the term $N_{2,x}$. For this we will finally make real use of the choice of our distribution F for choosing the variables p_i. First, using the definition of the expected value, we get

$$N_{2,x} = \frac{1}{\pi-4\delta'}\int_{\delta}^{1-\delta} \frac{p^x(1-p)^{n-x}\left(xq-\frac{n-x}{q}\right)}{\sqrt{p(1-p)}}dp. \tag{5.32}$$

Since $f(p) = p'$ if and only if $f^{-1}(p') = p$, with f^{-1} defined as $f^{-1}(p') = \sin^2(p')$, we can also get p by taking $r \in_R [\delta', \pi/2-\delta']$, and take $p = \sin^2(r)$. Then substituting r for p in the integral gives $1-p = cos^2(r), q = \sqrt{(1-p)/p} = \cos(r)/\sin(r), 1/q = \sin(r)/\cos(r), dp = 2\sin(r)\cos(r)dr$, so that we get

$$N_{2,x} = \frac{1}{\pi/2-2\delta'}\int_{\delta'}^{\pi/2-\delta'} \sin^{2x}(r)\cos^{2(n-x)}(r)\left(\frac{x\cos(r)}{\sin(r)} - \frac{(n-x)\sin(r)}{\cos(r)}\right)dr. \tag{5.33}$$

The primitive of the integrand is given by $I(r) = \frac{1}{2}\sin^{2x}(r)\cos^{2(n-x)}(r)$ so we get

$$N_{2,x} = \frac{I(\pi/2-\delta') - I(\delta')}{\pi/2-2\delta'} = \frac{\delta^{c-x}(1-\delta)^x - \delta^x(1-\delta)^{c-x}}{\pi-4\delta'}, \tag{5.34}$$

where we have used that $\sin^2(\delta') = \delta$ and $\cos^2(\delta') = 1 - \delta$. So for $0 < x < n$ we get

$$-\beta N_{2,x} \leq \frac{\beta\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} (> 0), \tag{5.35}$$

and

$$M_x \leq N_{0,x} + \frac{\beta\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} + \beta^2 N_{3,x}, \tag{5.36}$$

while for $x = 0$ we have $M_0 = N_{0,0}$ and for $x = c$ we have

$$M_n = N_{1,n} \leq N_{0,n} - \beta N_{2,n} + \beta^2 N_{3,n} = N_{0,n} + \frac{\beta\delta^n - \beta(1-\delta)^{c-x}}{\pi - 4\delta'} + \beta^2 N_{3,n}. \tag{5.37}$$

Now we look at the summation over M_x again, using the bounds we just found. Then

$$\sum_{x=0}^{c} \binom{c}{x} M_x \leq \sum_{x=0}^{c} \binom{c}{x} N_{0,x} - \frac{\beta(1-\delta)^c - \beta\sum_{x=1}^{c}\binom{c}{x}\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} + \beta^2 \sum_{x=0}^{c} \binom{c}{x} N_{3,x}. \tag{5.38}$$

For the summation over $N_{0,x}$, we use the fact that $\binom{c}{x}p^x(1-p)^{c-x}$ exactly counts the probability of having x successes in c trials, so that summing over all values of x, from 0 to c, gives exactly 1. So

$$\sum_{x=0}^{c} \binom{c}{x} N_{0,x} = \sum_{x=0}^{c} \binom{c}{x} \mathbb{E}_p[p^x(1-p)^{c-x}] = \mathbb{E}_p\left[\sum_{x=0}^{c} \binom{c}{x} p^x(1-p)^{c-x}\right] = \mathbb{E}_p[1] = 1. \tag{5.39}$$

For the summation over $N_{3,x}$ we get

$$\sum_{x=0}^{c} \binom{c}{x} N_{3,x} = \sum_{x=0}^{c} \binom{c}{x} \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(xq - \frac{c-x}{q}\right)^2\right] \tag{5.40}$$

$$= \mathbb{E}_p\left[\sum_{x=0}^{c} \binom{c}{x} p^x(1-p)^{c-x}\left(xq - \frac{c-x}{q}\right)^2\right]. \tag{5.41}$$

Note that for a specific x in this summation, the first part of the formula again calculates the probability of getting x successes in c trials. The latter part then calculates the accusation value S_j, as we saw before (count q for a success, count $-1/q$ for a failure). If we let U_j, $1 \leq j \leq c$, denote i.i.d. random variables with $\mathbb{P}[U_j = q] = 1 - \mathbb{P}[U_j = -1/q] = p$, then this last expectation is exactly equal to $\mathbb{E}[(\sum_{j=1}^{c} U_j)^2]$. Since the variables U_j are independent, and each has expectation 0 and variance 1, we have $\mathbb{E}[(\sum_{j=1}^{c} U_j)^2] = c$, so that

$$\sum_{x=0}^{c} \binom{c}{x} N_{3,x} = c. \tag{5.42}$$

Finally, for the term with $N_{2,x}$, we get

$$(1-\delta)^c - \sum_{x=1}^{c} \binom{c}{x} \delta^x(1-\delta)^x = 2(1-\delta)^c - \sum_{x=0}^{c} \binom{c}{x} \delta^x(1-\delta)^x = 2(1-\delta)^c - 1. \tag{5.43}$$

Since $(1-\delta)^c \geq 1 - c\delta$, we get $2(1-\delta)^c - 1 > 1 - 2c\delta$. So for the summation of M_x we finally get

$$\sum_{x=0}^{c} \binom{c}{x} M_x < 1 + \beta\left(\frac{2c\delta - 1}{\pi - 4\delta'} + \beta c\right). \tag{5.44}$$

Using $c \geq 1$, $\beta = \sqrt{\delta}/c$ and $\delta = 1/(300c)$ this can be further bounded to

$$\sum_{x=0}^{c} \binom{c}{x} M_x < 1 + \frac{\beta}{\pi}\left(-1 + 1/150 + \pi\sqrt{3}/30\right). \tag{5.45}$$

This then eventually gives

$$\sum_{x=0}^{c} \binom{c}{x} M_x < 1 - \frac{\beta}{4}. \tag{5.46}$$

So finally, for $\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right]$ we get

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x\right)^{\ell} < (1-\beta/4)^{\ell} < e^{-\beta\ell/4}. \tag{5.47}$$

So we can bound $\mathbb{P}[S \leq cZ]$ by

$$\mathbb{P}[S \leq cZ] \leq \frac{e^{-\beta\ell/4}}{e^{-\beta cZ}} = e^{-\beta(\ell/4 - cZ)}. \tag{5.48}$$

Using $\ell = 100c^2k$, $Z = 20ck$, $\beta = \sqrt{\delta}/c$, $\delta = 1/(300c)$ and $k = \lceil n/\epsilon_1 \rceil$ we get

$$\mathbb{P}[S \leq cZ] \leq \mathbb{P}[S \leq cZ] < e^{-k\sqrt{c}/4} \leq (\epsilon_1/n)^{\sqrt{c}/4}. \tag{5.49}$$

So the probability that no guilty user is accused is bounded by

$$\mathbb{P}[C \cap \sigma(\vec{y}) = \emptyset] < (\epsilon_1/n)^{\sqrt{c}/4}, \tag{5.50}$$

which concludes the proof. □

As mentioned, for Theorem 5.14 we will only give a sketch of the proof, and refer the reader to the extended version of [Tar03] for the complete version of the proof.

Sketch of the proof of Theorem 5.14. To make Theorem 5.13 work, one now has to take $\beta = \mathcal{O}(1/c)$ instead of $\beta = \sqrt{t}/c$ to end up with a factor c instead of $\sqrt{c}$ later on. Then the requirement $\beta(xq - (c-x)/q) < 1.79$ does not always hold anymore, so that $e^u \leq 1 + u + u^2$ does not always hold for $u = \beta(xq - (c-x)/q)$. To make the same proof work, one then has to add an extra term $\chi_x(p)\exp(\beta(c-x)/\sqrt{1-p})$ to the right hand side of $e^u \leq 1 + u + u^2$ for $u = \beta(xq - (c-x)/q)$, so that the inequality does hold. Continuing the proof, one then also has to bound the extra term

$$R_x = \mathbb{E}_p\left[\chi_x(p)(1-p)^{c-x}\exp(\beta(c-x)/\sqrt{1-p})\right]. \tag{5.51}$$

After some careful bookkeeping this term eventually vanishes, and one gets the same inequality $\mathbb{P}[S \leq cZ] \leq e^{-\beta(\ell/4 - cZ)}$ as in the proof of Theorem 5.13. However, then one can use $\beta = 1/(20c)$ so that $\mathbb{P}[S \leq cZ] < (\epsilon_1/n)^{c/4}$, as desired. □

5.4.2.5 Summary

Using the original Tardos scheme from [Tar03], we saw that we can guarantee security with a codelength of only $\ell = 100c^2\lceil\log(n/\epsilon_1)\rceil$, i.e. quadratic in c and logarithmic in $1/\epsilon_1$ and n. Although the constant 100 is big, this scheme is already a big improvement over the previous

scheme, the Boneh-Shaw scheme. Especially for large values of c, the quadratic term is a big improvement over the quartic term in the Boneh-Shaw scheme.

Although we hope the intuition behind the scheme became clear in the Introduction, the actual implementation of the scheme with exact values came out of the blue, with numbers like $\ell = 100c^2k$, $\delta = 1/(300c)$ and $Z = 20ck$ appearing without a clear explanation. These constants were chosen in such a way that the soundness and completeness properties can be proven, but why these parameters were exactly chosen like this is not clear. As it turns out, in many areas of this scheme there is also room for improvement, which we will investigate next.

5.4.3 Improvements

As we saw in the previous subsection, using certain parameter choices and using certain mathematical bounds in the proofs of soundness and completeness allowed Tardos to prove that his original fingerprinting scheme is ϵ_1-sound and ϵ_2-complete for $\epsilon_2 \geq \epsilon_1^{c/4}$. However, these parameters were mostly chosen as nice, round numbers, and no motivation for this choice was given. So maybe with a different choice of parameters, one can get a shorter codelength while still being able to prove soundness and completeness. Or one could simply try to tighten the proof, since some of the bounds used (such as the Markov inequality) are not tight. Below we will discuss some papers that used this approach to make the Tardos scheme even better, and further reduce the codelength.

5.4.3.1 Symmetric score function

One obvious improvement, which remarkably enough wasn't suggested until 2008, is to also use non-zero accusation weights when $y_i = 0$, i.e. define S_{ji} differently in those cases. In [SKC08] the following definition of S_{ji}, for $q = 2$, was introduced, which was also used in the Introduction.

$$S_{ji} = S_{ji}(y_i, (\vec{x}_j)_i, p_i) = \begin{cases} +\sqrt{(1-p_i)/p_i} & X_{ji} = 1, y_i = 1 \\ -\sqrt{p_i/(1-p_i)} & X_{ji} = 0, y_i = 1 \\ -\sqrt{(1-p_i)/p_i} & X_{ji} = 1, y_i = 0 \\ +\sqrt{p_i/(1-p_i)} & X_{ji} = 0, y_i = 0 \end{cases} \tag{5.52}$$

Using this definition of S_{ji} has several advantages. Most importantly, if all pirates receive a symbol 0, then in the original scheme their scores would not increase as y_i is 0. With this new definition, their scores increase a lot on these positions. In fact, out of the two cases where the marking assumption applies (all zeroes or all ones), in the original scheme we only made use of the case where all users had a one. Now we make use of both cases, which basically makes the scores go up twice as fast. This is also roughly the result from [SKC08], and they proved that using this symmetric Tardos scheme, the codelength can be reduced by a factor 4, compared to earlier results. This symmetric score function will also be used in Part II.

5.4.3.2 Optimizing the scheme parameters

One can investigate whether the choice of parameters was fully justified, or if we can use a better choice of parameters. As it turns out, also here we can get reasonable improvements compared to the original Tardos scheme.

In [SVCT06] the constants were all substituted by variables which were to be specified later. This resulted in some awful equations, e.g. equation (88) on page 21. Using numerical evaluations of this equation, one obtains better parameter sets satisfying all the necessary equations. Figure

3 in [SVCT06] shows that for $\epsilon_2 = \epsilon_1^{c/4}$, as in the original Tardos scheme, the codelength can be reduced by a factor between 1.1 and 1.2. Figures 1 and 2 in the same paper show that for the practical case that $\epsilon_2 \gg \epsilon_1$, we can reduce the codelength by a factor between 2 and 2.5, depending on the exact choices of ϵ_1, ϵ_2 and c. In any case, this definitely leads to big improvements as well.

In [BT08] a similar approach was used, by reconsidering the parameter choices in the original Tardos scheme. Most of the work in this paper is similar to the analysis in [SVCT06], but one important inequality which was not parametrized in [SVCT06] was parametrized here: $e^x \leq 1+x+x^2$ for $x < 1.79$. Instead this inequality was substituted by $e^x = 1+x+\omega(x)x^2$, where $\omega(x)$ is some function of x. This further tightens the bounds, and leads to some improvements compared to [SVCT06]. Especially for large c their results are good, decreasing the codelength from the original Tardos scheme by a factor 4. This improvement, together with the symmetric score function, is the basis for Chapter 8 in Part II.

5.4.3.3 Using the Gaussian approximation

Finally an important improvement, which however is harder to prove rigorously, is to apply the Central Limit Theorem to the accusation scores, as was done in [SVCT06], [SKC08] and is done in [SS10]. Recall that $S_j = \sum_{i=1}^{\ell} S_{ji}$ with ℓ reasonably large. For innocent users, we furthermore have that all the S_{ji} are i.i.d. random variables with mean 0 and variance 1. We therefore expect that S_j (almost) behaves like a normally distributed random variable with mean 0 and variance ℓ. If this is the case, we can simply calculate the probability that this user is falsely accused, i.e. that this normally distributed random variable exceeds some threshold Z.

Similarly, for the guilty users, we expect that the total coalition score S will behave like a normally distributed random variable as well, as ℓ goes to infinity. This is because pirates cannot really influence this total score. Using this Gaussian approximation of the distributions of S and S_j, the analysis becomes simpler, and we get another factor 2 improvement over the results in [SVCT06] and [SKC08].

5.4.3.4 Combining improvements

Using the results from the Gaussian approximation, the symmetric accusation weights and the optimized scheme parameters from [SVCT06], the paper [SKC08] shows that a codelength of $\ell = \frac{\pi^2}{2} c^2 \lceil \ln(n/\epsilon_1) \rceil \approx 4.93 c^2 \lceil \ln(n/\epsilon_1) \rceil$ is sufficient for secureness. This is under the assumption that the scores behave exactly like normal distributions, i.e. this only holds with certainty for infinite c.

If we only use the results from the symmetric accusation weights and the optimized scheme parameters, then we can show that a codelength of roughly $\ell = \pi^2 c^2 \lceil \ln(n/\epsilon_1) \rceil \approx 9.87 c^2 \lceil \ln(n/\epsilon_1) \rceil$ is sufficient for secureness for large c. This "large c" can then be specified exactly, i.e. if certain bounds are met, then with this codelength and certain parameters we get secureness with given maximum error probabilities.

5.4.4 Summary

After an extensive introduction, explaining the idea behind the scheme, we analyzed the original Tardos scheme and mentioned the improvements suggested in literature. The original Tardos scheme is already impressive, theoretically and practically, for achieving a codelength which is only a constant factor off from the theoretical lower bound.

Although this original scheme is already quite good, we saw that with some basic improvements one can do even better. Making some realistic assumptions, we can prove that for big c we can improve the original scheme by a factor 20. Then we are only a factor 3.61 off from the theoretical minimum of $2\ln(2)$. If however we want to be able to prove secureness and do not want to make such assumptions, then we can change the scheme to get codelengths that improve upon the original Tardos scheme by a factor between 2 and 20.

5.5 Summary

At the beginning of this chapter we looked at lower bounds on the codelength. In 2003 it was proven that the codelength has to be at least quadratic in c and logarithmic in n and ϵ_1. Some papers then showed that this bound is tight, and these papers investigated the exact constant for the codelength. Amiri and Tardos claimed to have solved this problem in [AT09] for asymptotically large c, giving the exact minimum codelength.

The first scheme we then investigated was the Boneh-Shaw scheme, which is intuitive and has a length polynomial in c. However, the original scheme had a codelength which was quartic in c, and [PSS03] showed that any Boneh-Shaw-like scheme that is secure needs to have a codelength which is at least cubic in c. This means that with the Boneh-Shaw scheme, we cannot approach the quadratic lower bound on the codelength.

Finally we looked at the Tardos scheme, which does achieve the quadratic lower bound, but has a relatively large constant. Using several improvements, this constant can be further reduced to narrow the gap between the required codelength in the Tardos scheme and the lower bound on the codelength of any scheme by a factor of less than 4. This scheme, together with these improvements, is currently the best probabilistic static scheme known in literature.

Chapter 6

Deterministic dynamic schemes

***Citations**: For writing this chapter, the following articles were used: [FT01], [BPS00]*

6.1 Introduction

In the previous chapters we considered static schemes using one single fingerprinting code and only one feedback moment, i.e. when the forgery is sent from the pirates to the distributor. Here we will look at dynamic schemes, which allow more interaction between the distributor and the users. In this chapter we will focus on deterministic dynamic schemes, which are dynamic schemes which trace traitors with no error, e.g. any accused user is known for certain to be guilty.

One can remark that a series of fingerprints, such as a series of forgeries received by the distributor or a series of fingerprints received by a user in a dynamic scheme, can be seen as one longer fingerprint. So one may ask the question whether using such a dynamic scheme with, say, t sequential fingerprints of length ℓ' in a dynamic scheme is different from using one fingerprint of length $\ell = t \cdot \ell'$ in a static scheme, and whether it leads to any improvements compared to static schemes. As we will see later in this chapter, the answer to both questions is affirmative: one can indeed get a better efficiency using dynamic schemes instead of static schemes. In particular, the theoretical lower bounds that hold for deterministic static schemes are violated by some of the best deterministic dynamic schemes. This can be explained by the fact that the distributor now has more freedom and more information when generating fingerprints, while pirates have less information, e.g. the pirates are forced to output forgeries before they know which symbols they will receive next.

As a first result, we show that there is still a strong relationship between dynamic and static schemes, and that the following result about IPP codes translates directly to a property of deterministic dynamic schemes.

Theorem 6.1 (Minimum alphabet size of deterministic dynamic schemes)**.** *[FT01, Theorem 1] To catch a coalition of c pirates using a deterministic dynamic traitor tracing scheme, we need an alphabet size of at least $c + 1$. Furthermore, this bound is tight: there exist algorithms to catch a coalition of c pirates using an alphabet of size $c + 1$.*

We will not prove this Theorem, since the proof is analogous to the one for IPP codes, e.g. see Theorem 4.41 and its proof on page 34 for details. In this case we do identify sequences of fingerprints with one longer fingerprint, i.e. identify deterministic dynamic schemes with IPP codes to prove this result.

We mention one more important result which says there is essentially no difference between catching one pirate or catching all pirates in this setting. This result allows us to be less precise about whether we write "catching one pirate in $\mathcal{O}(t)$ time" or "catching all pirates in $\mathcal{O}(t)$ time", since both are equivalent (assuming $t = \Omega(c)$, which generally is a safe assumption). This also allows us to be imprecise about when an algorithm should terminate; one may want to catch all pirates, or one could choose to save time by terminating when one pirate is caught.

Theorem 6.2 (Catching one or all pirates)**.** *[BPS00, Lemma 2.1] Let C be a coalition of size c. Then there exists a deterministic dynamic scheme that always catches at least one of the c traitors in t time if and only if there exists a deterministic dynamic scheme that traces all c traitors in $t + c - 1$ time.*

Proof. Obtaining an algorithm to trace one pirate in t time, given an algorithm that traces all c traitors in $t + c - 1$ time is trivial. If all c traitors are traced deterministically, then all c traitors must have been assigned some symbol σ_j at some time $t_j, 1 \leq j \leq c$, so that no other users received that same symbol at that same time, and the coalition outputted that specific symbol. Since only one traitor can be traced in one time step, we have that $t_j < t_{j+1}$ for all j, and $t_c \leq t + c - 1$. This means that $t_1 \leq t$, so that stopping after time t ensures that at least one pirate has already been caught.

Now suppose we have an algorithm A tracing a single traitor in t time. We now define an algorithm B, which uses algorithm A to trace all traitors in $t + c - 1$ time. For this, algorithm B simply runs algorithm A, by passing on symbol assignments and the symbols outputted by the coalition back and forth between algorithm A and the users, until at some moment a pirate is exposed and would be disconnected by algorithm A. Instead of passing on this symbol to A, algorithm B disconnects the pirate, and rebroadcasts the same symbols to all users except for the disconnected user, which is now out of the game. In such a step, the time B uses grows 1 larger than the time A uses. Now, if a new symbol was outputted, it is passed on to A, and the scheme continues. This is repeated until the penultimate pirate is disconnected by B, and the last pirate will be caught by A. By that time, B has run an extra $c - 1$ time steps compared to A, and by passing on the final symbol to A and letting algorithm A catch this last pirate, algorithm A found 1 pirate in t time and algorithm B found all pirates in $t + c - 1$ time. □

Note also the use of the word "algorithm" in this chapter. In the dynamic setting we no longer consider schemes as a pair of an accusation algorithm and a fingerprinting code, but as one big accusation algorithm using several shorter traitor tracing codes sequentially.

6.1.1 Graph notation

Before we start with the schemes and the lower bounds, we present the graph notation introduced in [BPS00] which reduces the problem of symbol distribution to a kind of vertex coloring problem.

Definition 6.3 (Graph notation for dynamic schemes)**.** *[BPS00, Section 2.3] Let the set of users U be partitioned into disjoint subsets $S_1, \ldots, S_k$, such that all users in some subset receive the same symbol. Then the graph describing the current state of the algorithm, denoted by Γ, is defined as $\Gamma = (V, E)$ with vertex set $V = \{S_i \mid 1 \leq i \leq k\}$ and edge set $E \subseteq \{S_i \sim S_j \mid S_i \cup S_j$ is known to contain a traitor$\}$ [1]. There is always one special vertex $I \in V$, containing "innocent users", i.e. users not known to contain a traitor. This vertex (usually) has degree 0. The assignment of symbols to subsets is respresented in this graph as a vertex coloring function*

[1] Note that there are no assumptions on the maximality of E. We assume that if there is an edge between vertices S, T, then it must be known that $S \cup T$ contains a traitor, but we do not assume the converse, i.e. that if it is known that $S \cup T$ contains a traitor, then S and T are connected in Γ.

$\phi : V \to Q$, where $\phi(S) = \omega$ means that the color ω (the symbol ω) is assigned to the vertex S (the users in subset S).

As an example, consider the following graph. The labels of the graph contain the numbers 1 up to 9, which indicates that there are 9 users in total. In this case, these users are partitioned into five sets, and the edges indicate the existence of a traitor in (at least) one of the two endpoints of the edge, e.g. the sets $\{4, 5, 6\}$ and $\{1, 2, 3, 8\}$ must both contain a traitor. Since any vertex cover in this graph has size at least 2 (e.g. $V' = \{\{5, 6\}, \{1, 3\}\} \subseteq V$ covers all edges) we know that there are at least two traitors assisting in creating a forgery. And since a vertex cover of size 2 exists, it could be that there are only two traitors, e.g. $C = \{1, 5\}$ satisfies the implications from the graph.

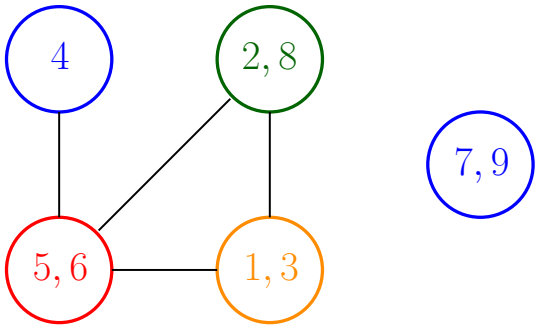

Figure 6.1: *An example of the graph coloring used in this chapter. For example users* 1 *and* 3 *get the color (symbol) orange, while users* $4, 7, 9$ *receive the color (symbol) blue. The edge between* $\{4\}$ *and* $\{5, 6\}$ *implies that the union* $\{4\} \cup \{5, 6\} = \{4, 5, 6\}$ *contains a pirate. Since both* $\{4, 5, 6\}$ *and* $\{1, 2, 3, 8\}$ *must contain at least one traitor, we know that there must be at least two traitors. A possible coalition is* $C = \{1, 5\}$*, as then all edges are covered by the vertices the two colluders are in.*

Since we want to make progress, by closing in on the traitors, there are typically two cases to consider when feedback is received from the pirates, and two common steps to take in those two cases. These are described below. In the main algorithms described in this chapter, other situations never arise.

The received color belongs to only one vertex.
In this case, the distributor knows that this subset contains a traitor, and a common action in the algorithm in that case is to split the corresponding vertex into two new vertices, each containing about half of the users in the original subset, and to add an edge between these two new vertices. One can add this edge, since it is known that the big subset, i.e. the union of the two smaller subsets, contains a traitor. The original vertex is removed from the graph, along with all its edges, and the two new connected vertices are inserted in the graph.

The received color belongs to two vertices, not connected by an edge.
In that case, the action to take in the graph is simple: add an edge between the two vertices. This edge is justified if only these two vertices received the output color, as then the union of these two sets must contain a traitor (which defines a justified edge).

After this change in the graph, one needs to reassign symbols to subsets of users, i.e. assign a new coloring to the vertices of the graph. After defining this new coloring, the distributor again waits for a response from the pirates, and after that repeats one of the two steps above. In the next sections we will give schemes and describe exact implementation details for updating

the graph and coloring the vertices in different schemes, and prove runtime results for these algorithms.

As we will be using this graph notation a lot, we will use different terminology in this chapter compared to the rest of the report. We will for example refer to different symbols as different colors, and call subsets of users vertices instead.

6.1.2 Example

Finally let us briefly consider a simple exponential time algorithm, suggested in [FT01], to illustrate the graph notation and describe how such algorithms work. This is a very simple and inefficient algorithm; it simply checks all subsets of c users to see if that subset is the coalition. This algorithm uses $c+1$ colors and runs in finite time, thus also proving that there exist deterministic dynamic schemes using only $c+1$ colors.

Construction 6.4. *[FT01, Section 3.2] Start with $t = 0, c' = 0, \Gamma = (V, E)$ with $V = \{I\}$, $I = U$, and $E = \emptyset$. Throughout the scheme always $E = \emptyset$, and every distinct vertex receives a distinct color.*

For every selection $u_1, \ldots, u_{c'}$ of c' users, set $I = U \setminus \{u_1, \ldots, u_{c'}\}$ and $V = \{\{u_1\}, \{u_2\}, \ldots, \{u_{c'}\}, I\}$. Let each vertex get a distinct color. Now if the pirates ever transmit a symbol belonging to a singleton vertex, we disconnect that user and decrease c' by one. Else if all selections of c' users have been checked and no user is disconnected, we increase c' by one and repeat this procedure for every selection of $c'+1$ users.

As mentioned, this algorithm basically tries all possible coalitions until the right coalition is found. This is done by assigning each of these c users a seperate symbol, and assigning the remaining users the same symbol. If the coalition is this set of c users, then they must output one of these symbols that are assigned to only one user. Then the distributor knows this user is guilty and disconnects him, and continues. If on the other hand the coalition can always output a symbol belonging to the remaining users, then the coalition size must be greater than our guess c', and we increase c' by one. Eventually the coalition is forced to output one of these special symbols, which will certainly happen for some $c' \leq c$. And since there are at most $c'+1$ colors used at any time, it follows that this construction uses at most $c+1$ colors at any point in time. For the running time, we get the following result.

Theorem 6.5. *[FT01, Section 3.2] Using Algorithm 6.4, we can catch all c colluders in at most $t = \mathcal{O}(n^c)$ time, with a codelength of $\ell = 1$ at each step.*

Proof. In the worst-case scenario, one has to increase c' all the way up to c until one pirate is caught. This takes at most $\binom{n}{c'}$ steps for each value of c', i.e. $\sum_{c'=0}^{c} \binom{n}{c'} = \mathcal{O}(n^c)$ time in total. So the running time to catch one or all pirates satisfies $t = \mathcal{O}(n^c)$. □

Of course this is a horrible running time. In the worst case, the total amount of data needed to catch the pirates is $t \cdot \ell = \mathcal{O}(n^c)$ symbols over an alphabet of size (at most) $c+1$. This algorithm merely serves as an example and it shows that we can in fact catch all pirates using only $c+1$ symbols (colors).

The rest of the chapter is divided as follows. First, in Section 6.2 we discuss theoretical lower bounds on the running time and total codelength of deterministic dynamic schemes. In Section 6.3 we will then discuss the Fiat-Tassa scheme, which has an optimal codelength for an alphabet size of (at most) $2c+1$. Then in Section 6.4 we will discuss several schemes which were described in [BPS00], most of which use only $c+1$ cymbols. Finally in Section 6.5 we review the results from this chapter.

6.2 Lower bounds

Similar to the previous chapter, we first try to find lower bounds on the effort needed to catch pirates. These provide us targets for what runtimes we would like to achieve using these schemes. In [BPS00, Section 7], the following lower bound was proven for deterministic dynamic schemes when no information about c known in advance.

Theorem 6.6. *[BPS00, Lemma 7.1] Let $\alpha \geq 1$ be some constant. Then to catch a coalition of size at most c, using an alphabet of size $q = c + \alpha$, at least $\Omega(c^2/\alpha)$ rounds are needed if no sufficiently small bound on the number of pirates c is known in advance.*

Proof. For $\alpha \geq c$ the bound obviously holds, since then to catch c traitors at least $\Omega(c^2/\alpha) = \Omega(c)$ rounds are needed. Now assume $\alpha < c$ and $n = c + \alpha + 1$, i.e. there are c colluders and $\alpha + 1$ innocent users. We now prove that one can keep outputting symbols in such a way that no user can be accused with certainty before some time t, when c is not known. This is done by proving that for any user j, the set $U \setminus \{j\}$ could have generated that same sequence of output symbols. In other words, it could theoretically be possible that user j was framed by all other users conspiring together. Only when t is sufficiently large (i.e. $t = \Omega(c^2/\alpha)$) it could be possible that one user is found guilty with absolute certainty.

We maintain a graph G, where every user forms a different vertex and there are initially no edges. Since $n > q$, by the pigeonhole principle, at any point in time there are two users receiving the same color. We will then output this color, and add an edge between the two users in our graph G. If more than two users received the same color, pick any two of them and add an edge between these two users.

Our first claim is that if there exists an independent set of size $\alpha + 1$ in G, then we can continue outputting symbols as above. Recall that an independent set is a set of vertices with no edges connecting any pair of these vertices. If there exists an independent set of size $\alpha + 1$ in G, then the answers are thus consistent with the remaining c users forming a coalition and these $\alpha + 1$ users being innocent. And as long as we can output symbols that are assigned to at least two users, there cannot be any accusations, if c is not known.

Now our second claim is that if G does not contain an independent set of size $\alpha + 1$, then G must contain at least $\Omega(c^2/\alpha)$ edges. To prove this, notice that an independent set of size $c + 1$ in G corresponds to a clique of size $c + 1$ in the complement graph of G, say H. By Turan's Theorem [Tur41], any graph on $n = c+\alpha+1$ vertices with no clique of size $r+1 = \alpha+1$ contains at most as many edges as the Turan graph $T(n, r)$, which contains at most $n^2/2 - n^2/(2r)$ edges. Since G and H together have $n^2/2 - n/2$ edges, the number of edges in G is at least $n^2/2 - n/2 - (n^2/2 - n^2/(2r)) = 1/2(n^2/r - n) = \Theta(n^2/r) = \Theta(c^2/\alpha)$, which proves the claim.

So by outputting symbols assigned to at least two users, no user can be accused with certainty. Above we showed that this can be done in a way consistent to some pirate coalition for at least $\Omega(c^2/\alpha)$ rounds, hence at least $\Omega(c^2/\alpha)$ rounds are needed to catch the coalition. □

If not only c is unknown but also n is significantly larger than c (which is usually the case), then the following Theorem gives an even stronger lower bound on the time needed to catch pirates. We will not prove this Theorem here.

Theorem 6.7. *[BPS00, Theorem 7.2] Let $\alpha \geq 1$ be some constant, and let $\lambda > 0$ be some constant such that $n \geq (1 + \lambda)c$. Then to catch a coalition of size at most c, using an alphabet of size $q = c + \alpha$, at least $\Omega(c^2/\alpha + c \log_{\alpha+1}(n))$ rounds are needed if no sufficiently small bound on the number of pirates c is known in advance.*

Theorems 6.6 and 6.7 have several interesting consequences. For example, taking $\alpha = 1$ in Theorem 6.6 tells us that if no information on c is known in advance, then any deterministic dynamic scheme will need at least $\Omega(c^2)$ time to catch the pirates. This is very similar to the static schemes we saw earlier; there also the lower bound was quadratic in c. In that sense this does not show how we get an improvement by going to the dynamic model. But if we take $\alpha = c$ in Theorem 6.6, then the lower bound only tells us that $t = \Omega(c)$. Then Theorem 6.7 gives a stronger lower bound, namely $t = \Omega(c \log_{\alpha+1}(n))$, but both bounds are only linear in c. Of course one could blame the lower bounds: perhaps the lower bound is indeed still quadratic in c, but we simply failed to show it here. However, this is not the case, as we will see later; taking $q = 2c + 1$, i.e. $\alpha = c + 1$, we can construct a scheme which is indeed only linear in c and logarithmic in n.

Finally note that we can make the logarithmic term in Theorem 6.7 as small as we want by taking α sufficiently large. If $\alpha = \mathcal{O}(n^{1/\beta})$ for some constant β, then the lower bound will simply be $t = \Omega(\beta c)$. In particular, $\beta = 1$ gives a bound linear in c. This is however trivial; if $\alpha = n$ then $q \geq n$ and we can simply give each user a different symbol, so that we trivially catch one pirate in every round.

6.3 The Fiat-Tassa scheme

6.3.1 Introduction

The first real dynamic scheme we will look at is the Fiat-Tassa scheme, introduced in [FT01] and also discussed in [BPS00]. This algorithm is the simplest of the ones still to follow in this chapter, and it is arguably the most useful one to follow as well, both for its simplicity and its efficiency. As this algorithm is basically an extended version of the **binary search algorithm**, we will first recall how this algorithm works, and in particular how it can be used here using our graph notation. The extension to the Fiat-Tassa scheme will then be natural and intuitive, and only requires a little additional work.

First of all, the binary search algorithm is a method which is most notably used for searching in sorted arrays. Given a sorted array A (i.e. $A[i] < A[j]$ for all $i < j$) and an element α known to be in that array, the problem is to find the index i of that element in the array (i.e. find i such that $A[i] = \alpha$). If the length of the array is n, then this can be done in $\log_2(n)$ time using the binary search algorithm. Starting with $a = 1$ and $b = n$, at every iteration step the invariant is that we know that $a \leq i \leq b$. At every step we then compare α with the middle element, i.e. if $m = \lfloor (a + b)/2 \rfloor$, then this element is $A[m]$. If $\alpha > A[m]$, then we set $a = m + 1$, while if $\alpha \leq A[m]$ we set $b = m$. In both cases, the interval size $b - a$ decreases by a factor 2. So in $\log_2(n)$ time, the interval size is decreased to 1 and we have found i.

The binary search algorithm relies on the fact that with one binary question at each step (i.e. Does $\alpha > A[m]$ hold?) we can eliminate half of the remaining suspects as suspects. In the above example this can be done because the array is sorted, which implies that if $\alpha > A[m]$ then α is not somewhere before position m. However, the applications are not limited to this specific instance, and the method works as long as we can ask any binary question that eliminates half of the space we are searching in. In particular it can be applied to tracing a single traitor as follows. We use three colors: one for each half of the space we are searching in, and one for the rest. We then distribute these colors, and wait for the pirate response. We then eliminate the part that had the wrong color, and add it to the section of innocent people. The users that did receive the same color as was broadcast by the pirate are again split into two groups, and the procedure repeats. Figures 6.3 and 6.4 show the process graphically.

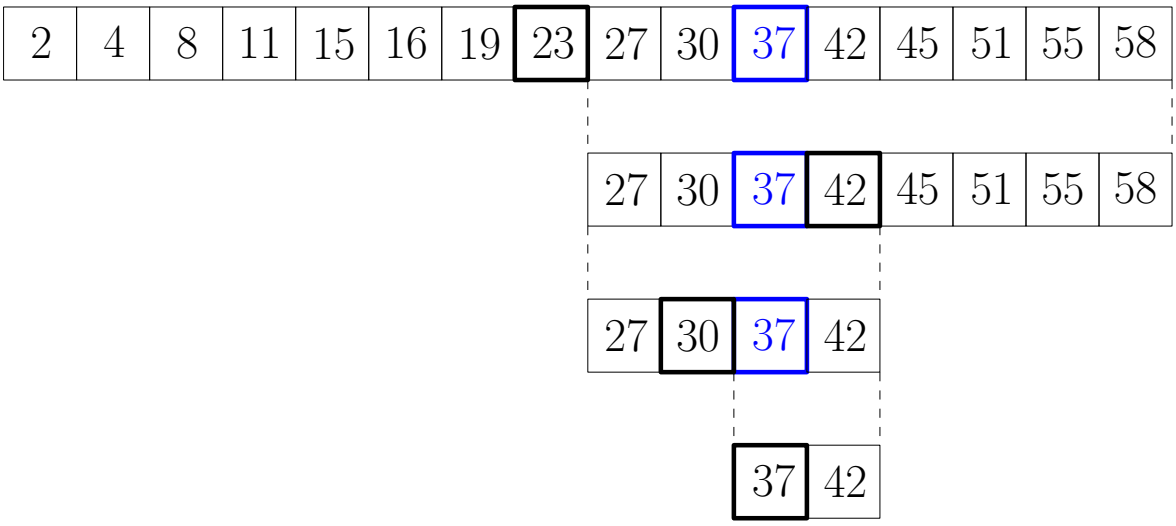

Figure 6.2: *The binary search algorithm applied to a sorted array A of length* 16*. The pivots are marked bold and black, while the number to be found is marked bold and blue. Finding the index of number* 37 *takes* 4 *steps, and in particular finding any number takes at most* $\log_2(16) = 4$ *steps.*

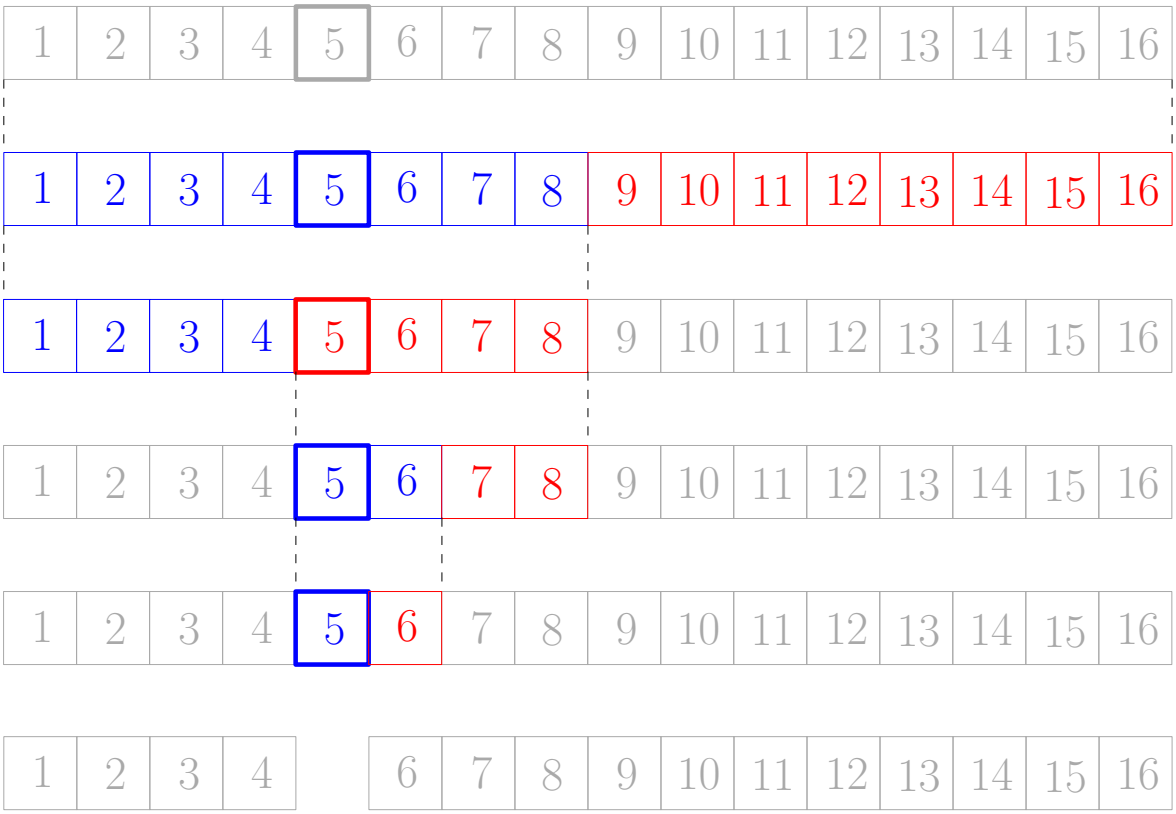

Figure 6.3: *The binary search algorithm used for tracing one traitor among* 16 *users, using only* 3 *colors. Finding the traitor, user* 5*, involves at most* $\log_2(16) = 4$ *steps of using more than* 1 *color. If we were to remove the gray numbers in the above figure, we would clearly get the same structure as in Figure 6.2.*

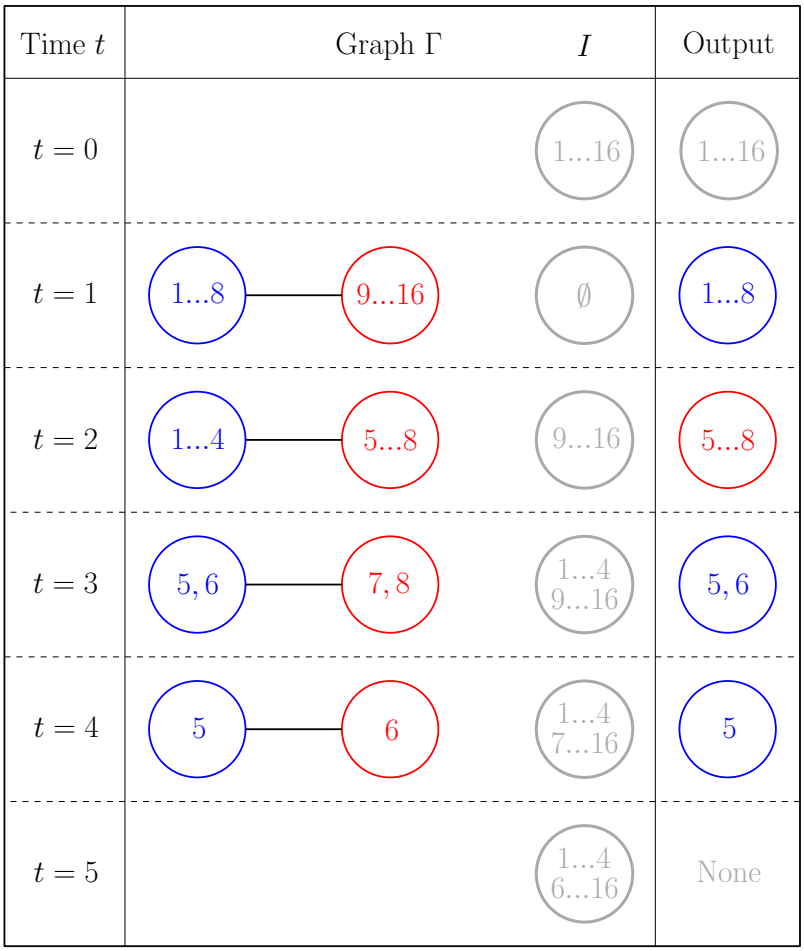

Figure 6.4: *The graph representation for the binary search algorithm as given above. There are always at most* 3 *vertices, and the size of the vertices not equal to* I *decreases by a factor* 2 *in every step.*

	$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
Alice	0	1	1	0	0	0
Bob	0	1	1	0	0	0
Charlie	0	1	1	0	0	0
Dave	0	1	1	0	0	0
Eve	0	1	2	1	1	-
Fred	0	1	2	1	2	0
George	0	1	2	2	0	0
Henry	0	1	2	2	0	0
Isaac	0	2	0	0	0	0
...	(0)	(2)	(0)	(0)	(0)	(0)
Olaf	0	2	0	0	0	0
Peter	0	2	0	0	0	0
Forgery	0	1	2	1	1	-

Table 6.1: *The symbol distribution for the previous example of tracing a single traitor among* 16 *users. The symbols distributed at time* t *depend on the values* y_i *for* $i < t$*. Eve is found guilty and disconnected after* 5 *symbols, and so she doesn't receive a* 6*th symbol.*

So using a slightly modified version of the binary search algorithm, we can find a single pirate with an alphabet of size only 3 in $\log_2(n)$ time.

6.3.2 The Fiat-Tassa scheme

While the above shows how to efficiently catch a single pirate, we are mostly interested in catching coalitions of multiple pirates. Then the scheme above no longer works, since at some point another pirate may jump in and distribute his symbol. Then we are already using two colors for the first traitor, and the third color, used for innocent users, is output by a pirate.

However, we can extend this binary search algorithm to catch multiple pirates, by running multiple binary searches simultaneously. First we use the same algorithm as above, with again the invariant that any edge between two vertices indicates that the union contains a pirate. And again, if the color received belongs to a vertex X connected to another vertex Y, then we add the users from Y to I and split X into X' and Y' of equal size, connected by an edge.

Now in the case of multiple pirates, we will likely run into the situation that at some point the received color belongs to the vertex I, i.e. one pirate is in the group of users that was presumed innocent until now. What we do then is we split this vertex I into two new vertices V and W of equal size and connect them by an edge. We make a new empty I, and we continue with 5 vertices instead of 3 and 5 colors instead of 3. We do this every time we discover there is a traitor in I; the number of vertices and colors then increases by 2, and we know the number of pirates is at least 1 more than we previously thought. In total this means we need at most $2c + 1$ vertices and colors, since there are always at most c cliques of size 2 and one vertex I. After some vertex has size 1 and its color is returned, we disconnect this single user, knowing that he is a pirate, and we therefore need two fewer colors afterwards.

This whole scheme was first described in [FT01], and in [BPS00] the associated graph notation was introduced. Formally, this leads to the following algorithm.

Construction 6.8. *[FT01, Section 3.3] Start with $t = 0$ and start with a graph $\Gamma = (V, E)$ with $I = U, V = \{I\}, E = \emptyset$, and set $c^* = 0$. All vertices are always given distinct colors. Let X denote the vertex belonging to the color received by the distributor at some time t. Then for each round t, do the following:*

1. *If $|X| = 1$, then the single user in X, say u, must be a traitor. Disconnect user u and set $c^* = c^* - 1$.*
   (a) *If $X = I$, set $I := \emptyset$ and set $c^* = c^* + 1$.*
   (b) *If $X \neq I$, then X is connected to some vertex Y. Add Y to I, and remove X and Y from the graph.*
2. *If $|X| > 1$, then we can split X into two non-empty subsets X' and Y', containing $\lceil |X|/2 \rceil$ and $\lfloor |X|/2 \rfloor$ users respectively. Add the vertices X', Y' and the edge $X' \sim Y'$ to the graph.*
   (a) *If $X = I$, update $I := \emptyset$, and set $c^* = c^* + 1$.*
   (b) *If $X \neq I$, then X is connected to some vertex Y. Add Y to I, and remove X and Y from the graph.*

In the construction, c^* stands for a lower bound on the active pirates, i.e. if $c^* = 5$ then there are at least 5 pirates still active. Note that the steps for $X = I$ and $X \neq I$ are the same in both cases; if $X = I$, then we increase c^* as the output does not correspond to any of the c^* pirates we know to be active. If $X \neq I$ and $X \sim Y$, then we add Y to the set of 'innocent' users, and

depending on whether $|X| = 1$ or $|X| > 1$, we either found a pirate, or we get a new pair of smaller vertices $X' \sim Y'$.

As an example, we applied the construction to the case of 16 users and two pirates, namely users 3 and 12. Figure 6.5 shows the result when putting the users in an array, while Figures 6.6 to 6.16 show the same example with our graph notation.

For verifying the scheme, we simply mention some of the invariants of this algorithm, which can easily be verified.

1. For all vertices $X \in V$, either $X = I$ or X is connected to some unique vertex $Y \in V$.
2. If $X \in V$ is connected to some $Y \in V$, then $X \cup Y$ contains a traitor.
3. The number of active pirates in the coalition is at least c^*, hence $c \geq c^*$.
4. At any moment in time the number of vertices satisfies $|V| = 2(c^*) + 1 \leq 2c + 1$.
5. At any time the number of edges is given by $|E| = c^*$.

For the runtime and the fact that indeed all pirates are caught, first note that we only disconnect guilty users. Since there are only c of these, the first option in the construction can only be encountered c times. In the second case, where $|X| > 1$, we split X into two new vertices X' and Y'. If $X = I$, then c^* is increased, which can only be done at most c times. If $X \neq I$, then Y is added to I and we replace X by the two new vertices. These are half the size of X and Y, hence the size of this pair of vertices decreases by a factor 2. For every pair of vertices this can thus only occur at most $\log_2(n)$ times, after which both vertices have size 1. And since there are c such pairs, this can occur $c \log_2(n)$ times in total. Summing up these runtimes for each possible option, we get the following result.

Theorem 6.9. *[FT01, Theorem 2] For any c, the Fiat-Tassa algorithm traces all c pirates in at most $c \log_2(n) + c$ time, using at most $2c + 1$ colors.*

6.3.3 Summary

The Fiat-Tassa scheme is basically an extension of the binary search algorithm. A binary search takes $\log_2(n)$ time to find one item, and by running c of these searches simultaneously the Fiat-Tassa scheme can find all c traitors in at most $\mathcal{O}(c \log_2(n))$ time. The cost for this is that we need two colors (symbols) per traitor and one additional color for the innocent users, i.e. at most $2c + 1$ colors in total. So in terms of traitor-tracing schemes, we need an alphabet of size $2c + 1$ and $t = \mathcal{O}(c \log_2(n))$ time (and $\ell = 1$ at each step) to trace all traitors.

Looking at the static schemes we have seen before, we see that this is actually the first scheme which is linear in c (and logarithmic in n). Both deterministic and probabilistic static schemes have to satisfy lower bounds which are quadratic in c. While we already saw earlier that we could not find a lower bound quadratic in c, we now see the reason why. The sharpest lower bound we saw in Theorem 6.7 for an alphabet size of at most $2c + 1$ is $\mathcal{O}(c \log_c(n))$, so this scheme is almost tight. Asymptotically we are only a factor $\log_2(c)$ short.

The price we pay for the low running time is that we need an alphabet size of at most $2c+1$. Since we are working with deterministic accusations we need $q > c$, but there is still a big difference between $q = c + 1$ and $q = 2c + 1$. This could make this scheme impractical, even though the runtime is very low. In the next section we will investigate constructions from [BPS00] which use only $c + 1$ colors.

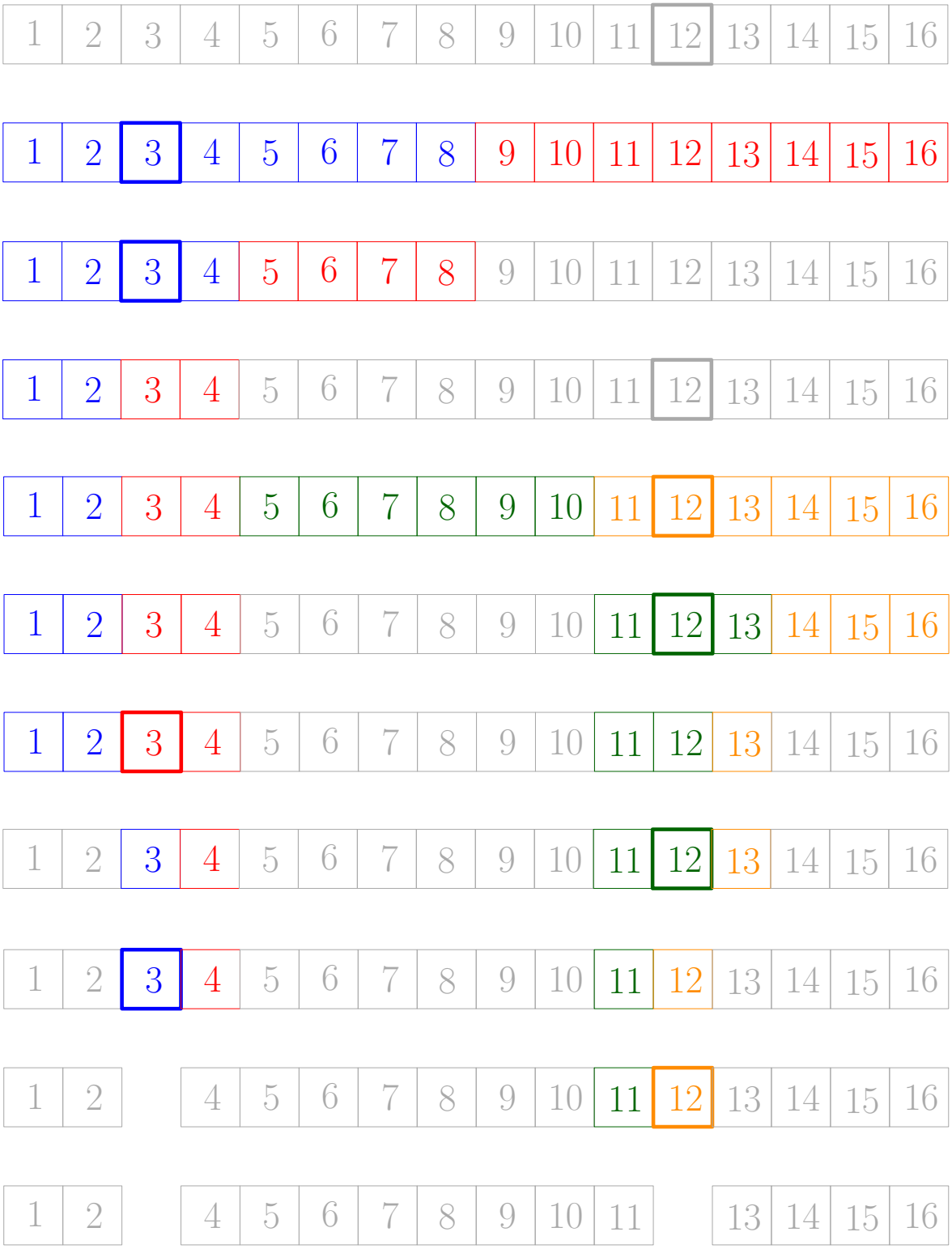

Figure 6.5: *The Fiat-Tassa algorithm as given above for tracing two colluders. There are always at most* $2c + 1 = 5$ *vertices (colors) in the graph, and the time needed to isolate both traitors is roughly* $2\log_2(16) = 8$*. The colors red and blue are used for tracing the first traitor, while orange and green are used for the second traitor. The color gray is used for the special vertex* I*. At the end both pirates are disconnected.*

Figure 6.6: *There are* 16 *users in the system, which are all initially added to* I.

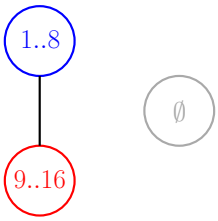

Figure 6.7: *We received output from user* 12*, so* $c \geq 1$ *and we split* I *into two new vertices. We add a new empty vertex* I.

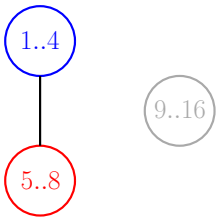

Figure 6.8: *This time the color received by user* 3 *was received, so this vertex is split and its neighbor is added to* I.

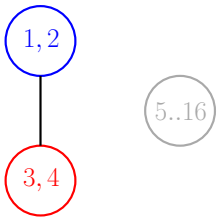

Figure 6.9: *Again user* 3 *outputs his symbol, so that we again split this vertex in halves and add the neighbor to* I.

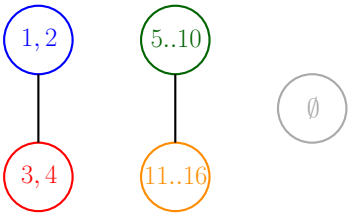

Figure 6.10: *User* 12*, who was in* I*, distributed his symbol, so* c^* *is increased to* 2 *and we now use* 5 *vertices and colors.*

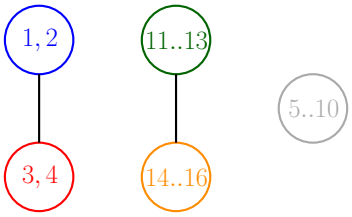

Figure 6.11: *Again user* 12 *distributes his copy, so his vertex is split into two and its neighbor vertex is added to* I.

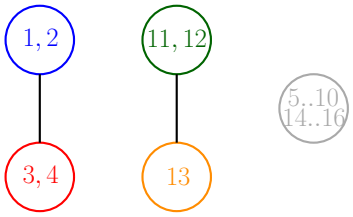

Figure 6.12: *The colluders again choose to output the symbol assigned to user* 12*, resulting in another split.*

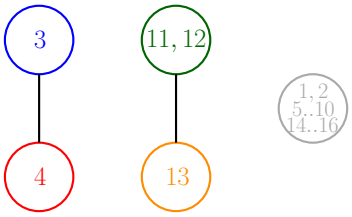

Figure 6.13: *Now pirate* 3 *publishes his color, so his vertex is split in halves, and the users* $1, 2$ *are again added to* I.

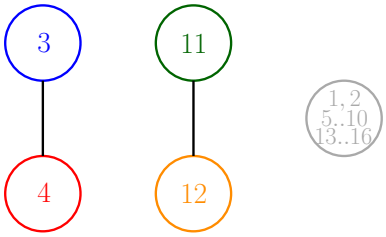

Figure 6.14: *The pirates can get away with publishing their symbol one last time, as user* 12 *is not yet isolated. After this step both are isolated, and so in two more iterations both will be caught.*

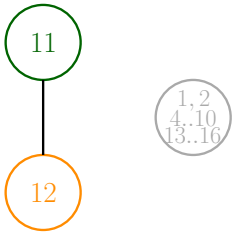

Figure 6.15: *No more splits are needed, since the only user to receive the distributed color was user* 3*. He must therefore be guilty and he is disconnected from the system. Now* c^* *decreases to* 1 *and we use only* 3 *colors from now on.*

Figure 6.16: *The last pirate, user* 12*, also output his symbol. He is captured and disconnected, and we set* $c^* = 0$*. From here on we only use color, as we suspect there are no pirates anymore.*

6.4 The Berkman-Parnas-Sgall schemes

6.4.1 Introduction

Shortly after Fiat and Tassa published their paper, containing the Fiat-Tassa scheme above, Berkman, Parnas and Sgall published a paper further investigating the concept of dynamic traitor tracing. In particular, this latter paper focused on the case of using the minimal number of colors for which a deterministic scheme exists, i.e. $q = c + 1$. The paper suggested several schemes, three of which we shall discuss below.

Besides the algorithms, the same paper also investigated lower bounds, which we already saw in Section 6.2. For $q \leq c + 1$ at any time and c unknown, Theorem 6.6 says we need at least $\Omega(c^2)$ time to trace traitors. If also n is really larger than c (i.e. $n > (1 + \lambda)c$ for some positive λ) then the lower bound from Theorem 6.7 says we need at least $\Omega(c^2 + c \log_2(n))$ time to catch pirates.

First we shall discuss an intuitive algorithm that runs in $\mathcal{O}(c^3 \log(n))$ time. Then we will discuss a scheme which also runs in $\mathcal{O}(c^3 \log(n))$ time, but which can be used and improved to finally form a scheme running in $\mathcal{O}(c^2 + c \log(n))$ time. This latter scheme is known as the optimal algorithm in [BPS00], as it asymptotically achieves the lower bound of Theorem 6.7.

6.4.2 The degree algorithm

First of all, we discuss the last algorithm from [BPS00], namely the degree algorithm. This algorithm is arguably the easiest one to understand from the algorithms described in this paper, which is why we explain this one first.

The idea behind the algorithm is as follows. We maintain a graph similar to the Fiat-Tassa scheme, with cliques of size 2 and one special vertex I. In this graph however we will have more edges than just these edges connecting the pairs of vertices. In the degree algorithm we further have only $c + 1$ colors instead of $2c + 1$, so we assign certain pairs of vertices (which are not connected in any way) the same color. If a color belonging to only one vertex is received, then we again simply split this vertex in two and add an edge inbetween, as in the Fiat-Tassa scheme. And in case the output color belongs to two vertices, we simply add an edge between the two vertices.

The important observation we use for this algorithm is that if some vertex has $c + 1$ neighbors, i.e. some vertex is connected to $c + 1$ other vertices, then this vertex itself must contain a traitor. After all, if this vertex does not contain a traitor, then because of these edges, these $c + 1$ neighbors in the graph each contain at least 1 traitor, giving a total of at least $c + 1$ traitors, which is a contradiction. So the idea of adding edges to the graph is to increase the degrees of vertices in the graph, such that eventually vertices will have too high degree to contain no traitors.

One problem we run into is that we do not know c and during the Fiat-Tassa algorithm we only know a lower bound c^* on the number of traitors. If some vertex has a degree of more than c^*, then this vertex only contains a pirate provided that $c^* = c$. Therefore if some vertex has degree more than c^*, we instead place this vertex in a waiting room. This vertex then stays there until either (a) we have verified that $c = c^*$, or (b) we have established that $c > c^*$, or (c) the unique color assigned to this vertex is output by the colluders. In cases (a) and (c) we would then split this vertex and insert it back into the graph, while if (b) is the case, we insert the vertex back into I and start over with c^* increased by one.

Besides using a waiting room, we now have to maintain two estimates of c: we write c^* for a lower bound on the number of traitors (and the current estimate for the total number of traitors),

and we write $\hat{c}$ for a lower bound on the number of traitors in distinct vertices of G, given that $c^* = c$. The variable c^* can be compared to the same variable in the Fiat-Tassa scheme. The variable $\hat{c}$ is used for determining if c^* is a correct guess. An invariant for the algorithm is that $\hat{c} \leq c^*$, so if at some point we get that $\hat{c} > c^*$ (which can occur if our guess c^* for c was wrong), then we know that $c \geq c^* + 1$, so that we can increase our guess c^* by one. The vertices that were in the waiting room are then added back to the main area and into the special vertex I, while in the main area no further edges or vertices are removed or changed. We then repeat the whole procedure, but with a better lower bound c^*.

Finally we also write $\hat{c}_1$ for the contribution to $\hat{c}$ from the main area, and $\hat{c}_2$ for the contribution to $\hat{c}$ from the waiting room, i.e. $\hat{c} = \hat{c}_1 + \hat{c}_2$. Thus in the main area we have $2\hat{c}_1 + 1$ vertices, according to the Fiat-Tassa scheme, and in the waiting room we have $\hat{c}_2$ vertices, each suspected to contain a traitor.

6.4.2.1 Color distribution

One problem we still have to solve is that in the main area we have $2\hat{c}_1 + 1$ vertices, and some complex network of edges between the vertices, and that we want to assign these colors to (pairs of) vertices that are not connected. And since we want to assign every vertex in the waiting room a distinct color, we only have $\hat{c}_1 + 1$ colors for these $2\hat{c}_1 + 1$ vertices. For this we need assurance that we can always find such pairs of unconnected vertices. If we investigate what this actually means, we see that these unconnected pairs of vertices correspond to connected vertices in the complement graph (i.e. $v \sim w$ in the complement of G if and only if $v \not\sim w$ in G). Then we can also translate these pairs to a matching in this complement graph, consisting of $\hat{c}_1$ edges. For this we can then use the following Lemma.

Lemma 6.10. *[BPS00, Lemma 6.1] Let H be a graph with at least $2m$ vertices, each having degree at least m. Then there exists a matching in H of size at least m.*

Proof. We prove that given a matching M of size $m^* < m$ in this graph H, we can make a matching of size $m^* + 1$ in this same graph. Starting with $m^* = 0$, by induction it then follows we can find a matching of size at least m in at most m steps.

Let v and w be two vertices in H which are not matched by M, and let N_v and N_w denote the sets of neighbors or v and w respectively. If v is connected to w in H, we can simply add $v \sim w$ to the matching and we are done. Also if some $x \in N_v \cup N_w$ is not matched by M, then we can trivially improve the matching by adding this edge $v \sim x$ or $w \sim x$. So if $M = \{x_i \sim y_i \mid i = 1, \ldots, m^*\}$ for some x_i, y_i, then we can assume that $N_v, N_w \subseteq \{x_1, y_1, \ldots, x_{m^*}, y_{m^*}\}$ and $v \not\sim w$, as otherwise we would already be able to improve the matching. We now prove that $|N_v \cap \{x_i, y_i\}| + |N_w \cap \{x_i, y_i\}| \leq 2$ for all i, so that $|N_v| + |N_w| = \sum_{i=1}^{m^*} |N_v \cap \{x_i, y_i\}| + |N_w \cap \{x_i, y_i\}| \leq 2m^*$. This is then a contradiction with $|N_v|, |N_w| \geq m$, i.e. $|N_v| + |N_w| \geq 2m > 2m^*$, proving the result.

Let x_i and y_i be two vertices matched to eachother by M. If $|N_v \cap \{x_i, y_i\}| + |N_w \cap \{x_i, y_i\}| > 2$, then w.l.o.g. we can assume $|N_v \cap \{x_i, y_i\}| = 2$ and $|N_w \cap \{x_i, y_i\}| \geq 1$, i.e. $v \sim x_i, y_i$ and $w \sim x_i$. But then we can improve the matching by removing the edge $x_i \sim y_i$ and adding the edges $v \sim y_i$ and $w \sim x_i$ to M, increasing the size of the matching by 1. This proves the result. □

Lemma 6.10 shows that with $m = c^* - \hat{c}_2 \geq \hat{c}_1$ we can always find a color assignment of $c^* - \hat{c}_2 + 1$ colors to the $2\hat{c}_1 + 1$ vertices in the main area such that no two vertices that are connected receive the same color. Adding the $\hat{c}_2$ colors used for the vertices in the waiting room, we get a total of at most $c^* - \hat{c}_2 + \hat{c}_2 + 1 = c^* + 1 \leq c + 1$ colors.

6.4.2.2 Graph reorganization

The most important thing we have not yet described in full is how exactly the graph is updated following the pirates' response. As most of the scheme has already been explained above, we do not go into detail in every step again, and simply give the algorithm.

Algorithm 6.11. *[BPS00, Section 6.2] Start with $t = 0$ and start with a graph $\Gamma = \Gamma_1 \cup \Gamma_2$, where $\Gamma_1 = (V_1, E \cup F)$ corresponds to the main area and $\Gamma_2 = (V_2, \emptyset)$ to the waiting room. The set E corresponds to the Fiat-Tassa edges, consisting of $\hat{c}_1$ distinct edges, while the edges from F are the edges added to the graph when two non-connected vertices receive the same color and that color is output. Initially $V_1 = \{I\}, I = U, E = \emptyset, F = \emptyset, V_2 = \emptyset$. Set $c^* = \hat{c} = \hat{c}_1 = \hat{c}_2 = 0$. For each round t, we do the following.*

1. *If only one vertex $X \in \Gamma_i$ received the output color, and $|X| = 1$, then the user in this vertex is guilty. Disconnect this user and set $c^* := c^* - 1, \hat{c}_i := \hat{c}_i - 1$.*
   (a) *If $X \in \Gamma_1$ and $X = I$, set $I := \emptyset$ and set $c^* = c^* + 1$.*
   (b) *If $X \in \Gamma_1$ and $X \neq I$, then X is connected to some vertex Y in E. Add Y to I, and remove X and Y from the graph.*
   (c) *If $X \in \Gamma_2$, set $\hat{c}_1 := \hat{c}_1 + 1$ and $\hat{c}_2 := \hat{c}_2 - 1$.*
2. *If only one vertex $X \in \Gamma_i$ received the output color, and $|X| > 1$, then we can split X into two non-empty subsets X' and Y', containing $\lceil |X|/2 \rceil$ and $\lfloor |X|/2 \rfloor$ users respectively. Add the vertices X', Y' to the graph, and add the edge $X' \sim Y'$ to E. Set $\hat{c}_i := \hat{c}_i - 1$ and set $\hat{c}_1 := \hat{c}_1 + 1$.*
   (a) *If $X \in \Gamma_1$ and $X = I$, update $I := \emptyset$, and set $\hat{c}_1 := \hat{c}_1 + 1$ and $c^* = c^* + 1$.*
   (b) *If $X \in \Gamma_1$ and $X \neq I$, then X is connected to some vertex Y in E. Add Y to I, and remove X and Y from the graph.*
   (c) *If $X \in \Gamma_2$, set $\hat{c}_1 := \hat{c}_1 + 1$ and $\hat{c}_2 := \hat{c}_2 - 1$.*
3. *If two non-connected vertices $X, Y \in \Gamma_1$ received the output color, add the edge $X \sim Y$ to F and do the following.*
   (a) *Any vertex $Z \in \Gamma_1$ of degree larger than $c^* - \hat{c}_2$ is removed from Γ_1 and placed into Γ_2. We then set $\hat{c}_2 := \hat{c}_2 + 1$ and if $Z = I$, we create a new vertex $I = \emptyset$.*
   (b) *Repeat the above until all vertices have degree at most $c^* - \hat{c}_2$. Note that the value of $c^* - \hat{c}_2$ decreases after placing a vertex Z into Γ_2, which may cause other vertices to have a too high degree now.*
   (c) *After all vertices have degree at most $c^* - \hat{c}_2$, the graph Γ_1 is messed up. Add all vertices which no longer have a neighbor in E to I, and update $\hat{c}_1$ according to the new number of two-cliques in Γ_1.*
4. *If after any of the above steps $\hat{c} > c^*$, we set $c^* := c^* + 1$ and add all vertices from Γ_2 to I.*

Some of the actions taken during this algorithm can be compared to the Fiat-Tassa algorithm, where the same steps are taken. Basically we semi-disconnect vertices which have a too high degree by placing them in the waiting room until we know whether (1) we were right and the vertex contains a traitor, or (2) we were wrong and c^* was a bad estimate, in which case we increase c^* by one. If we were right and the vertex contains a traitor, we take the same action as in the Fiat-Tassa algorithm, by splitting the vertex in two and adding it as a new clique of size two in the main area. Figures 6.17 to 6.32 give an example of how the algorithm makes progress.

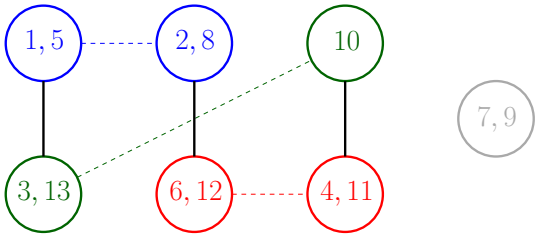

Figure 6.17: *There are* 13 *users in the system, and using a Fiat-Tassa-like graph we have established that there are at least* $c^* = 3$ *traitors. Therefore we use only* 4 *colors for the* 7 *vertices. At this point there is no waiting room, and* $\hat{c}_1 = 3, \hat{c}_2 = 0$.

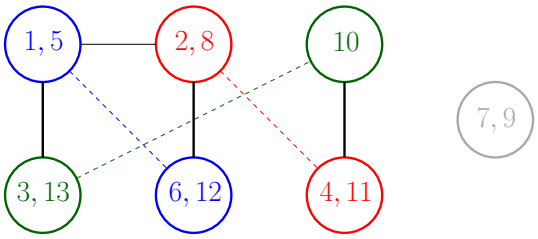

Figure 6.18: *After the color given to users* $1, 2, 5$ *and* 8 *was output, we added an edge and constructed a new coloring of the vertices with* 4 *colors. We again use a matching of size* 3 *on the complement graph of the main area.*

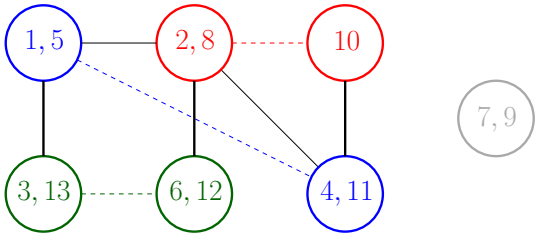

Figure 6.19: *The color given to users* $2, 4, 8, 11$ *was output, and another edge has been added to the graph. We still suspect there are* 3 *traitors.*

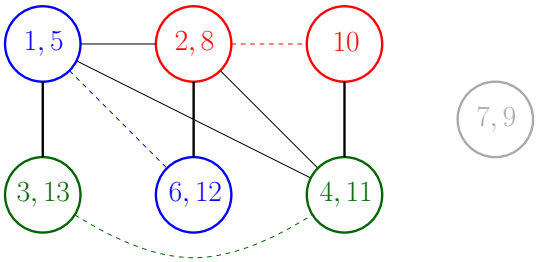

Figure 6.20: *Another edge was added to the graph, but all vertices still have degree at most* 3. *And we can still find a good coloring of the vertices.*

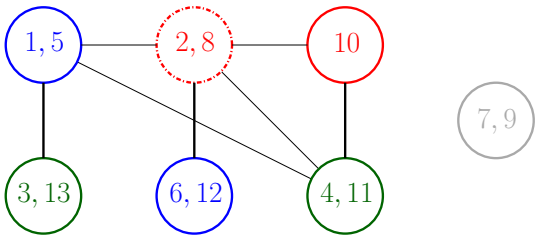

Figure 6.21: *After another edge was added, the degree of the vertex with users* 2 *and* 8 *is now higher than* 3, *i.e.* 4. *So this vertex will be moved to the waiting room, and its neighbor will be added to* I.

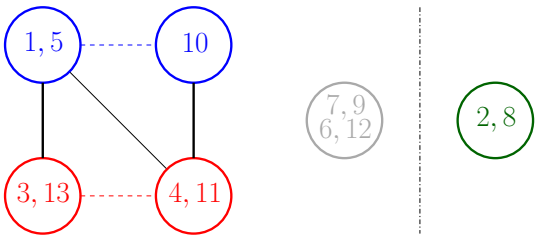

Figure 6.22: *After the reorganization, the suspected number of traitors in the main area has reduced to* 2. *All vertices there have a degree of at most* 2, *hence no further reorganization is needed. Now* $\hat{c}_1 = 2$ *and* $\hat{c}_2 = 1$.

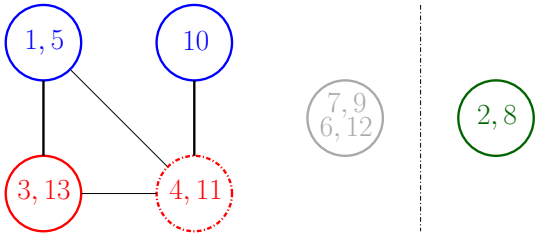

Figure 6.23: *The process continues with another edge added. Now vertex* $\{4, 11\}$ *has degree* 3, *which is more than* $\hat{c}_1 = 2$. *Therefore this vertex will be moved to the waiting room.*

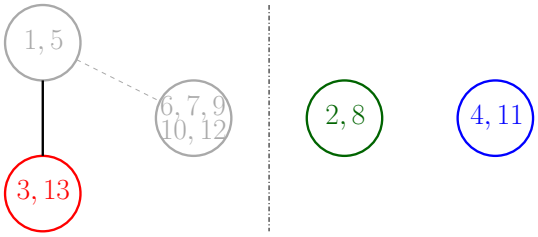

Figure 6.24: *After the reorganization, the suspected number of traitors in the main area has reduced to* $\hat{c}_1 = 1$. *However, all vertices have a degree of at most* 1, *hence no further reorganization is needed.*

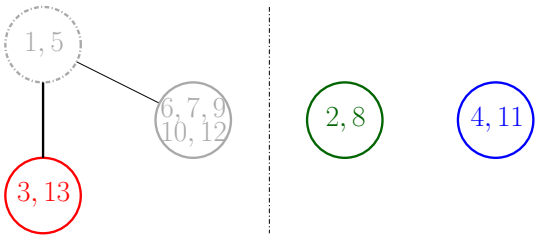

Figure 6.25: *Another edge was added, after which vertex* $\{1, 5\}$ *has degree* 2*. Since at this point* $\hat{c}_1 = 1 < 2$ *(and* $\hat{c}_2 = 2$*), this vertex will also be moved to the waiting room.*

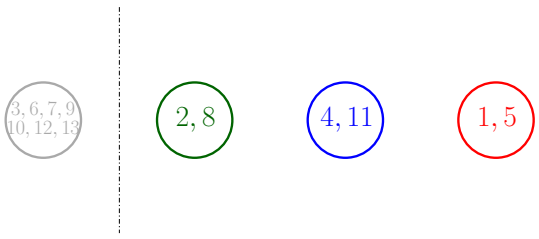

Figure 6.26: *Now we have that* $c^* = \hat{c}_2 = 3$ *and* $\hat{c}_1 = 0$*, i.e. if* $c = 3$ *then the three vertices in the waiting room each contain a traitor while the main area contains no traitors.*

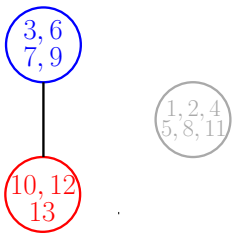

Figure 6.27: *The pirates have output the color gray, associated to vertex* I*. Our guess that* $c = c^* = 3$ *was therefore wrong:* c *must be at least* 4*. The vertex* I *was split as usual, and the vertices in the waiting room were added to* I*.*

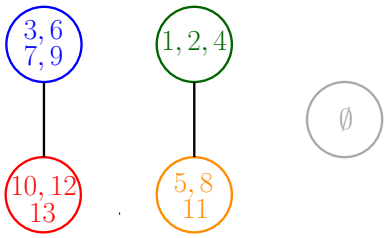

Figure 6.28: *The color gray was output again, so* I *is again split in two. Since we know that* $c \geq 4$*, we use* 5 *colors, i.e. one for each vertex. Here for the first time this fifth color is used.*

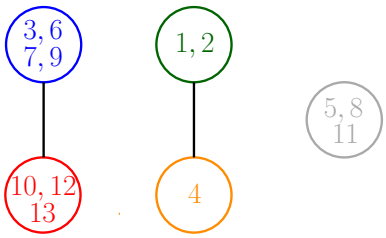

Figure 6.29: *The pirates again output the color gray, associated to the vertex* I*. We split this vertex, as in the Fiat-Tassa scheme, and create a new empty vertex* I*.*

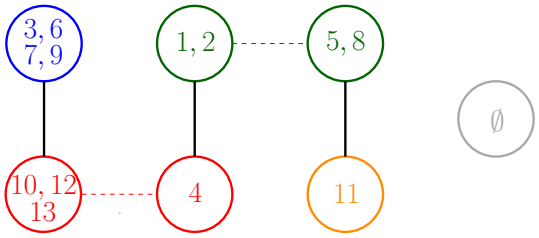

Figure 6.30: *Since we know that* $c \geq c^* = 4$*, we can use* 5 *colors to give each vertex a distinct color. Doing this is not necessary, but using the maximum number of colors available does decrease the running time slightly.*

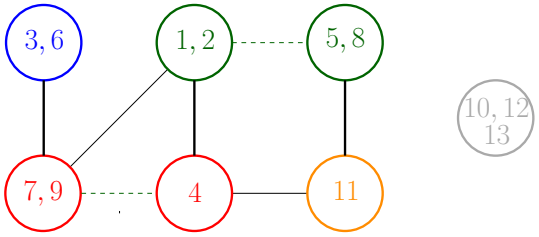

Figure 6.31: *After some more steps we again have* 7 *vertices. Some more edges have been added to the graph, since we were required to give pairs of vertices the same color again, and these colors were output.*

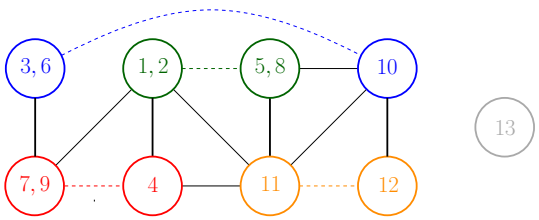

Figure 6.32: *Finally we end up with a graph that looks very similar to the start, but with* 4 *pairs of vertices instead of* 3*. The process continues after this, until all pirates are disconnected.*

Theorem 6.12. *[BPS00, Theorem 6.3] The degree algorithm traces any coalition of any size c in at most $\mathcal{O}(c^3 \log_2(n))$ time using at most $c+1$ colors.*

Proof. For the amount of colors used, see Subsubsection 6.4.2.1 on the color distribution in this algorithm, where it is shown we only need at most $c^* + 1$ colors at any time. As c^* is at most c at any time in the algorithm, we only need at most $c + 1$ colors.

For the time needed, first note that the value of c^* is always increased by at least 1, when it is increased. Thus proving that for every c^* between 0 and c the time taken is at most $\mathcal{O}(c^2 \log_2(n))$ proves the result.

If c^* is not increased at some point in time, then either (1) a vertex was split (or the single user in it is disconnected), or (2) a vertex is moved to the waiting room, or (3) an edge is added to the graph. If there are more vertices in the waiting room than c^*, then c^* is increased by one. Therefore, for a single value of c^*, only $c^* = \mathcal{O}(c)$ vertices can be moved to the waiting room. However, when splitting a vertex we may move a vertex from the waiting room back to the main area. Since a vertex is always split in two roughly equal-sized halves, the number of times a vertex can be split is $\mathcal{O}(\log_2(n))$. So only $\mathcal{O}(c \log_2(n))$ times can vertices be moved from and to the waiting room for a fixed value of c^*.

Finally there is the case of adding edges. We may add lots of edges before moving vertices from one area to another. However, in the main area there are always at most $2c + 1$ vertices, hence at most $\mathcal{O}(c^2)$ edges. When a vertex is moved to the waiting room, all edges incident with this vertex and all edges incident with its Fiat-Tassa neighbor are removed from the graph, thus removing at most $\mathcal{O}(c)$ edges. After splits, vertices may be moved back to the main area, when again we can start adding edges until we reach the limit. As there are at most $\mathcal{O}(c \log_2(n))$ splits, vertices are removed from the main area at most $\mathcal{O}(c \log_2(n))$ time, thus removing at most $\mathcal{O}(c^2 \log_2(n))$ in total. Hence there are at most $\mathcal{O}(c^2 + c^2 \log_2(n))$ edges added to the graph and at most $\mathcal{O}(c \log_2(n))$ steps involve vertices being moved from or to the waiting room. Thus in total we get at most $\mathcal{O}(c^2 \log_2(n))$ time needed for each value of c^*, proving the result. □

The running time of this algorithm is cubic in the number of colluders and logarithmic in the number of users. Compared to the Fiat-Tassa scheme this is of course very bad, with an additional factor of c^2 added to the running time. The gain is that we now only use an alphabet of size at most $c + 1$ rather than $2c + 1$. This however is only a factor 2 improvement, so the scheme is only useful if the alphabet size is a really big issue while the running time is not. Moreover we disregarded the constants in the running time here, while for the Fiat-Tassa scheme the constant is roughly 1 in the worst case.

6.4.3 The clique algorithm

We now move on to a different algorithm, suggested in [BPS00, Section 3]. Berkman et al. call it the clique algorithm, as it is based on getting large cliques rather than getting vertices with high degrees, as in the degree algorithm. An extension of the clique algorithm later also leads to their optimal algorithm in [BPS00, Section 4], which (almost) achieves an optimal asymptotic running time.

In this report we will not give an elaborate analysis of the algorithm. The bookkeeping done in the algorithm is quite complicated and is only necessary to make sure the details check out. The idea behind the algorithm however is easier to sketch and explain, which we will do. Therefore our explanation will leave certain issues unsolved and untreated. For these details we refer the reader to the original paper, [BPS00], which does discuss these details. For completeness we

do give the full algorithm as in [BPS00], but without the detailed explanation of the zones and blocks as in [BPS00] one may not completely understand the algorithm.

First of all, as the name suggests and as we mentioned a few lines earlier, the clique algorithm is based on getting large cliques of connected vertices. The first reason we want large cliques is simple: any k-clique contains at least $k-1$ pirates. This can be proven in one line[2]: If there are at most $k-2$ pirates, then some two of these k vertices do not contain a pirate, which contradicts the fact that they are connected by an edge. The second reason is perhaps even more obvious: we can easily color a k-clique with only k colors. This is also a 'good' coloring, i.e. no three vertices get the same color, and no two connected vertices get the same color. Therefore we can again split single vertices and add an edge between pairs of non-connected vertices once we receive the pirate output, as we also did with the Fiat-Tassa algorithm and the Berkman-Parnas-Sgall degree algorithm. And since we use only at most k colors for a k-clique containing $k-1$ pirates, the pirates-per-color rate goes up towards 1 as k increases. So bigger cliques are much appreciated.

Now how exactly does the algorithm work? Well, instead of running the Fiat-Tassa algorithm with connected pairs of vertices, we basically run the algorithm with the pairs of vertices replaced by pairs of cliques. Instead of having disjoint cliques of 2 vertices, we now have disjoint blocks, each containing 2 cliques. These cliques may be interconnected in some way, but the only thing we really need is that these cliques are really fully connected cliques.

6.4.3.1 Color distribution

For the colors, we assign a subset of the colors to each block of two cliques. If the two cliques, say Q_a and Q_b, contain a and b vertices respectively, then we know that there are at least $(a-1)+(b-1) = a+b-2$ traitors in this block. To use only as many colors as there are traitors, we therefore assign $a+b-2$ colors to this block. Now suppose $a > b$. For a good coloring we have to give all vertices in Q_a a different color, which is also what we do. For Q_b we then have only $b-2$ unused colors left. We color $b-2$ vertices with a new, different color, and color the remaining two vertices with two of the first a colors, in such a way that no two interconnected vertices get the same color. For this we need that there exist vertices $X_1, X_2 \in Q_a, Y_1, Y_2 \in Q_b$ such that $X_1 \not\sim Y_1$ and $X_2 \not\sim Y_2$. This is nearly always the case; only when also the cliques Q_a and Q_b are (almost) fully connected inbetween this will give problems. But if Q_a and Q_b are fully connected, or if some subset $\hat{Q}_b \subset Q_b$ is fully connected with Q_a, then we in fact have a bigger clique and the algorithm would have taken appropriate actions already.[3]

6.4.3.2 Graph reorganization

Using this color assignment, we again perform the same actions as in the Fiat-Tassa algorithm and the degree algorithm. If the color output by the pirate belongs to two non-connected vertices, then we add an edge inbetween. If some bigger cliques arise, we then also have to reorganize the graph, to make sure we can (1) continue with another good coloring in the next round, and (2) make progress as well. The algorithm starts with small cliques, and 'progress' is either adding more edges, replacing smaller cliques by bigger cliques or splitting/disconnecting a vertex.

[2]The actual number of lines in this report depends on the formatting, but the proof consists of only one reasonably short sentence which could easily fit on just one line.

[3]This 'appropriate actions' is something that is fully explained in the original paper, but which we will not discuss here. When such a larger clique shows up, the graph has to be reorganized, which involves a lot of bookkeeping. In the end we then again have blocks of two cliques, but one of the cliques has increased in size, which is the progress we made.

Algorithm 6.13. *[BPS00, Section 3.2] Start with $t = 0$ and start with a graph $\Gamma = (V, E)$. Initially $V = \{I\}, I = U, E = \emptyset$. Set $c^* = 0$ and start with one block, consisting of only the clique I. For each round t, we do the following.*

1. *If only one vertex X received the output color, and $|X| = 1$, then the user in this vertex is guilty. Disconnect this user and set $c^* := c^* - 1$.*
   (a) *If $X = I$, set $I := \emptyset$ and set $c^* = c^* + 1$.*
   (b) *If $X \neq I$ and X was in a clique Q of size 2, then X was in a clique with some vertex Y. Add Y to I, and remove X and Y from the graph. The other clique R that was in the same block as Q is now seperately incorporated into the graph.*
   (c) *If $X \neq I$ and X was in a clique Q of size at least 3, then removing X still leaves a clique of size at least 2. Therefore no further changes are needed.*
2. *If only one vertex X received the output color, and $|X| > 1$, then we can split X into two non-empty subsets X' and Y', containing $\lceil |X|/2 \rceil$ and $\lfloor |X|/2 \rfloor$ users respectively.*
   (a) *If $X = I$, update $I := \emptyset$, and set $c^* = c^* + 1$. Incorporate the new clique $Q' = X' \sim Y'$ into the graph.*
   (b) *If $X \neq I$ and X was in a clique Q of size 2, then X was in a clique with some vertex Y. Add Y to I, and remove X and Y from the graph. The new clique $Q' = X' \sim Y'$ will take the place of the clique $Q = X \sim Y$ in the graph.*
   (c) *If $X \neq I$ and X was in a clique Q of size at least 3, then removing X still leaves a clique of size at least 2. Seperately incorporate the new clique $Q' = X' \sim Y'$ into the graph.*
3. *If two non-connected vertices X, Y received the output color, add the edge $X \sim Y$ to E and do the following.*
   (a) *If adding the edge connects I to a clique Q in such a way that $Q \cup I$ now forms a clique, then we add I as a vertex to the clique Q, and we create a new vertex $I' = \emptyset$. The cliques Q and I' now again form a block.*
   (b) *If adding the edge connects two cliques Q and R in such a way that $Q \cup R \setminus \{Z\}$ forms a clique for some Z, then we create a new clique $Q' = Q \cup R \setminus \{Z\}$. The vertex Z is added to I, and the clique Q' is seperately incorporated into the graph.*
4. *If a new clique Q' needs to be incorporated into the graph, and I is already in a block with another clique Q, then we create a new block consisting of Q and Q', and we let I form a seperate block of only one clique.*
5. *If a new clique Q' needs to be incorporated into the graph, and I is not in a block with another clique, then we create a new block consisting of I and Q'.*

Using this algorithm, we get a scheme which catches any coalition using at most $c + 1$ colors. An example is again given in Figures 6.33 to 6.40. For the running time we get the following result.

Theorem 6.14. *[BPS00, Theorem 3.3] The clique algorithm traces any coalition of any size c in at most $\mathcal{O}(c^3 \log_2(n))$ time using at most $c + 1$ colors.*

For the running time, we do not give a proof, but we only mention a few things about the proof. The factor $\log_2(n)$ again comes from splitting vertices into two in every step. Furthermore, a factor c comes from the fact that (as in the degree algorithm) at most $\mathcal{O}(c)$ edges are removed when a vertex is split or some reorganization is done. Finally the remaining factor $\mathcal{O}(c^2)$ comes from the fact that in the worst case we have to create larger cliques many times before a vertex is split. In the end this then gives a total of at most $\mathcal{O}(c^3 \log_2(n))$ edges that can be added (and removed) during the algorithm, giving a running time of $\mathcal{O}(c^3 \log_2(n))$.

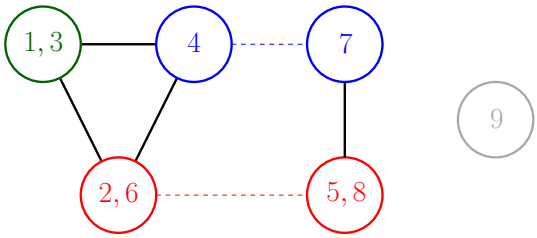

Figure 6.33: *There are* 9 *users in the system, and somehow we have obtained two cliques; one of size* 3 *and one of size* 2*. Therefore* $c^* = 3$ *and we can use* 4 *colors.*

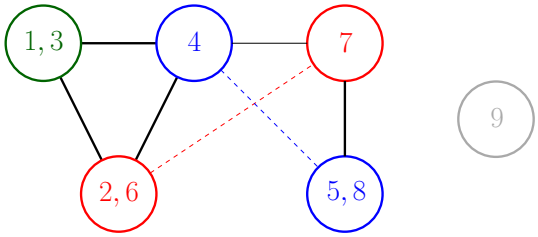

Figure 6.34: *The color given to users* 4 *and* 7 *was output by the pirates. An edge is added, and a new coloring using only* 4 *colors is constructed.*

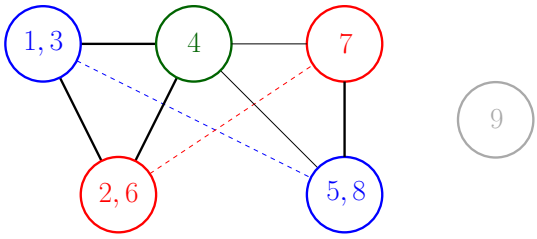

Figure 6.35: *The pirates now output the color assigned to users* $4, 5$ *and* 8*. We add another edge, and we construct a new coloring. There is no larger clique than the triangle yet.*

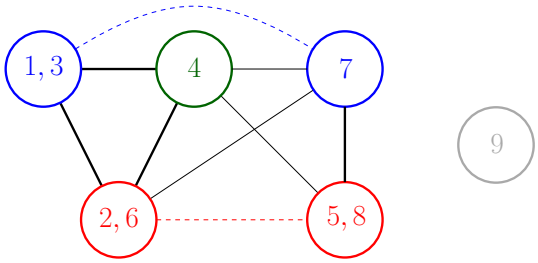

Figure 6.36: *Another edge is added to the graph, after the colluders distributed the color given to user* $2, 6, 7$*. Still there is no bigger clique than the triangle, and we continue with another new color assignment.*

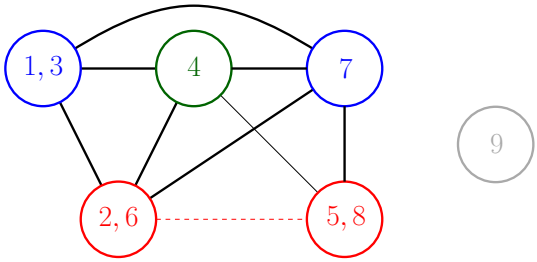

Figure 6.37: *Finally after another edge has been added to the graph, we see that vertex* $\{7\}$ *is connected to all vertices from the other clique. This means that we now have a larger clique of size* 4*.*

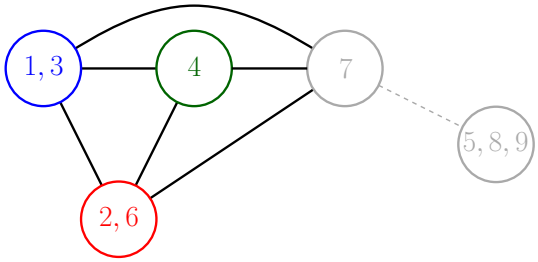

Figure 6.38: *The graph is reorganized: vertex* $\{7\}$ *is now part of the bigger clique, while its clique-neighbor* $\{5, 8\}$ *is added to* I*. The large clique will now form a special block with* I*.*

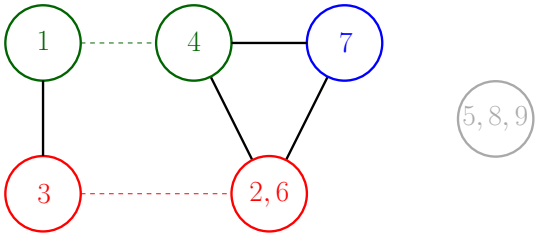

Figure 6.39: *This time the output color belonged to only one vertex, namely* $\{1, 3\}$*. Similar to the other algorithms, we simply split this vertex into two smaller ones and added an edge inbetween. This* 2*-clique then formed a new block with the clique of size* 3*.*

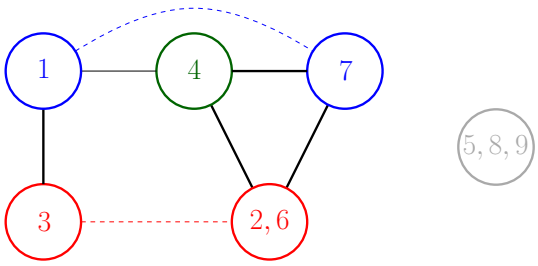

Figure 6.40: *Similar to the start, we again start adding edges when the distributed color belong to pairs of vertices. It may look like we did not make progress since the start, but we did: there are now less users in the clique-vertices and more users in* I*.*

6.4.4 The optimal algorithm

Besides the degree algorithm and the clique algorithm, which both run in $\mathcal{O}(c^3 \log(n))$ time, the paper [BPS00] also presented an algorithm running in only $\mathcal{O}(c^2 + c\log(n))$ time. This matches the lower bound of 6.7, which is why it is called the optimal algorithm in [BPS00]. However, compared to this optimal algorithm, the analysis from the previous section is peanuts; even in the journal version, Berkman et al. used over 10 pages to explain the optimal algorithm. This includes using 4 zones (i.e. not just Γ_1 and Γ_2 as in the degree algorithm, but $\Gamma_1 \dots \Gamma_4$), several blocks of cliques for each zone (the clique algorithm is used as a starting point for this scheme), and new marks to mark vertices which have a certain property. Especially the fourth zone is terribly complicated, for which one main algorithm and four subalgorithms are given to prove the correctness of the claims made about this zone.

Therefore we choose to only give the results about this scheme as claimed by Berkman et al. This can be summarized in one Theorem as follows.

Theorem 6.15. *[BPS00, Theorem 3.3] The optimal algorithm traces any coalition of any size c in at most $\mathcal{O}(c^2 + c\log_2(n))$ time using at most $c+1$ colors.*

Finally we note that Berkman et al. mentioned in [BPS00, Section 7] that the hidden constants in the running time are large. Adding to that the complexity of the algorithm, one could say this algorithm is really only useful for large values of c. Even then, one could argue that for such large values of c, the c^2 in the running time plays a larger role than using c more symbols for the alphabet (as we have to use a ridiculously large alphabet anyway). So then one may also argue that using the Fiat-Tassa scheme, with small constants, low complexity and only $\mathcal{O}(c\log(n))$ time, is better, even though then one needs an extra c symbols.

6.4.5 Summary

While in [FT01] Fiat and Tassa only mentioned algorithms for $q = c + 1$ with running times exponential in c, the paper [BPS00] proposed several schemes with running times polynomial in c. The first schemes we discussed here had a running time which is cubic in c, while we mentioned that another scheme was given with a running time only quadratic in c (without $\log(n)$) and linear in c with a factor $\log(n)$. The same paper also presented lower bounds for such schemes which we discussed earlier, and this optimal algorithm asymptotically matches the best lower bound up to a constant factor.

For practical reasons however, this optimal algorithm is not really optimal; the hidden constants are large, and the algorithm is very complex. The degree algorithm and the clique algorithm however have smaller hidden constants and a lower complexity. These schemes are intuitive and quite easy to follow, and are easier to analyze. The price we pay for that is a higher running time.

So if one is looking for deterministic schemes (in a dynamic setting) with a small-as-possible alphabet size, then this section provides good solutions; both simple ones (degree algorithm, clique algorithm) and complex but more efficient ones (optimal algorithm).

6.5 Summary

In this chapter we saw how we can efficiently trace traitors deterministically in a dynamic setting. As the tracing process is deterministic, the minimum alphabet size is $c+1$, and for this alphabet size Berkman et al. gave three efficient schemes in [BPS00]. Two schemes are relatively simple,

but require a runtime which is cubic in c and logarithmic in n. In that sense the third is much better, with an asymptotic runtime of $\mathcal{O}(c^2 + c\log(n))$, but this one is highly complex and has large hidden constants. This third scheme however matches the asymptotic lower bound, which is why this scheme is called the optimal algorithm.

As Berkman's lower bounds show, once we switch to an alphabet size of $q = c + \mathcal{O}(c)$ the lower bound on the runtime changes to $\mathcal{O}(c\log(n))$. And indeed, Fiat and Tassa showed in [FT01] that using an alphabet of size $2c + 1$ we can trace all traitors in only $\mathcal{O}(c\log(n))$ time. The constant in this asymptotic runtime is also roughly 1, so this scheme is really efficient for catching any size coalitions. The downside is however the large alphabet needed.

So the two papers gave solutions to two different problems. Berkman et al. showed how to efficiently trace a coalition with a minimum alphabet size, while Fiat and Tassa showed how to get an even better construction, using a bigger alphabet.

Chapter 7

Probabilistic dynamic schemes

***Citations**: For writing this chapter, the following articles were used: [Tas05]*

7.1 Introduction

In this last chapter of the literature survey we investigate the final class of schemes considered in this report, namely dynamic schemes that are probabilistic and allow a small error probability. Similar to the previous chapter, here we again consider the scenario where neither c, nor an upper bound on c is known in advance. This chapter will be considerably shorter than the earlier chapters, mainly because only one paper actually considered these types of schemes.

Note that a general principle associated to deterministic/probabilistic schemes also applies here. While for (both static and dynamic) deterministic schemes we needed an alphabet size of at least $c + 1$, for (static and dynamic) probabilistic schemes we only need an alphabet size of at least 2.

7.2 The Tassa scheme

First we recall that the Fiat-Tassa scheme is a deterministic dynamic scheme to trace all colluders in $\mathcal{O}(c \log(n))$ time, using at most $q = 2c + 1$ symbols. For details we refer the reader to Section 6.3, but here we only use the fact that at every time step, the associated graph contains $2c + 1$ vertices, and each vertex gets a different color. So at any time step, the set of users is partitioned into $2c + 1$ disjoint subsets, and each subset receives some symbol σ_i, for $i = 1, \ldots, 2c + 1$.

Next we recall that the cubic Boneh-Shaw scheme is a probabilistic static scheme to trace one user of any coalition (i.e. of any size c of at most n) with a codelength of $\mathcal{O}(n^3 \log(n/\epsilon_1))$. The scheme uses a binary alphabet, and has a maximum error probability of $\epsilon_1 = \epsilon_2$. In other words: all n users receive a codeword of length $\mathcal{O}(n^3 \log(n/\epsilon_1))$, and after receiving output from the pirates, the tracer accuses one of the users, which is part of the coalition with probability at least $1 - \epsilon_1$.

Using these two schemes, Tassa came up with the following construction for a probabilistic dynamic scheme. First, we simply apply the Fiat-Tassa scheme with its associated graph as the (outer) scheme. However, instead of using $q = 2c + 1$ symbols, we replace the symbols by codewords from the (inner) Boneh-Shaw scheme. So in each time step of the Fiat-Tassa scheme we replace a single symbol by a whole Boneh-Shaw codeword. After receiving the forgery from the pirates, the Fiat-Tassa scheme then performs one single step (e.g. splitting a vertex or disconnecting a user).

Intuitively, one can see that the outer scheme then still requires $\mathcal{O}(c\log(n))$ time steps. For the inner scheme, we see that there are always at most $2c+1$ subsets, and by identifying these subsets as users in the Boneh-Shaw scheme we see that we need a Boneh-Shaw scheme with $n \leq 2c+1$. Thus the codelength for the inner code is roughly $\mathcal{O}(c^3\log(c/\epsilon_1))$. Without any additional changes, at every time step we have a maximum error probability of ϵ_1. Therefore in the worst case the success probability at the end is $(1-\epsilon_1)^{\mathcal{O}(c\log(n))} = 1-\mathcal{O}(c\log(n))\epsilon_1$.

Using some further calculations on the probability that the scheme terminates, this is also roughly Tassa's result in [Tas05, Theorem 1]: Using this construction, the probability of finishing in at most $\mathcal{O}(c^4\log(c/\epsilon_1)\log(n))$ time is at least $1-\epsilon_1$, and the probability of not disconnecting any innocent users in this process is at least $1-\mathcal{O}(c\log(n))\epsilon_1$.

After this, Tassa proposes rescaling some of the variables in the scheme so as to be able to bound the false positive probability from above by ϵ_1, instead of $\mathcal{O}(c\log(n))\epsilon_1$. This results in [Tas05, Theorem 2], which has roughly the same asymptotic codelength/time, but has the error probability bounded by ϵ_1.

Note that although the paper [Tas05] was published in 2005, Tardos' better probabilistic static scheme was published only after [Tas05] was accepted for publication, as Tassa notes in [Tas05, Section 3]. For the inner code Tassa used the cubic Boneh-Shaw code, which at the time was the best known code for this purpose. However one can easily get better results using this same construction, by using the Tardos code as the inner code. This would simply save a factor c in the asymptotic codelength/runtime.

7.3 Summary

What stands out most about this chapter is its length. To our knowledge only one paper investigated what we classify as probabilistic dynamic schemes, namely the paper from Tassa in 2005. In this paper a construction was given by concatenating the Fiat-Tassa scheme as the outer scheme and the Boneh-Shaw scheme as the inner scheme. The resulting scheme has the runtime of the Fiat-Tassa scheme ($t = \mathcal{O}(c\log(n))$), and at each time step it has the same codelength as the Boneh-Shaw scheme for $n = 2c+1$ users (so $\ell = \mathcal{O}(c^3\log(c/\epsilon_1))$). In total this scheme therefore requires $t\cdot\ell = \mathcal{O}(c^4\log(c/\epsilon_1)\log(n))$ fingerprinting positions per user before the whole coalition is caught.

Besides using the Tardos code as the inner code instead of the Boneh-Shaw code, it is clear that there is still a lot of research to be done in the area of probabilistic dynamic fingerprinting schemes. Both chapters on deterministic fingerprinting schemes are based on tens of papers, some of which also give lower bounds to match or approach the best known constructions. For deterministic dynamic schemes as in Chapter 6 we only considered two papers, but together these contain four interesting constructions and some lower bounds matching the constructions, which show that a lot of progress has been made in that area already. For this chapter however we have only one paper containing one somewhat elementary construction and no lower bounds, while this area should be the easiest; the tracers get more feedback inbetween because of the dynamic scenario, and because of the probabilistic setting the scheme is even allowed to make small errors.

So to end this chapter, we conclude that the area of probabilistic dynamic schemes is still very much unexplored, and undoubtedly there is room for improvement here.

Part II

The Tardos Quadrilogy

Chapter 8

The optimal Tardos scheme

8.1 Introduction

As we saw in Chapter 5, the Tardos scheme is one of the most efficient traitor tracing schemes. The Tardos scheme uses a small alphabet, it requires only few and simple computations, and it has an (asymptotic) optimal codelength of order $\mathcal{O}(c^2 \log(n/\epsilon_1))$. It is therefore often considered the most practical static scheme for tracing traitors, if not the most practical overall scheme for tracing traitors.

While we extensively discussed the original Tardos scheme from [Tar03] in Chapter 5 and we only briefly mentioned improvements for this scheme, here we will focus on some of these improvements. We will investigate how we can obtain the best parameters for the Tardos scheme, while still maintaining provability for the soundness and completeness properties. By combining the ideas from [SKC08] (a symbol-symmetric accusation function) and [BT08] (making the analysis as tight as possible) we obtain a construction which achieves better results than both schemes separately. In fact, the results are optimal for this construction (with this distribution function and score function), which follows from a result from Skoric et al. [SKC08]

This chapter is organized as follows. In Section 8.2 we first give the construction of the symmetric Tardos scheme, and compare our results with earlier results from literature. In Sections 8.3 and 8.4 we then prove that the soundness and completeness properties hold under our assumptions on the parameters. In Section 8.5 we then give results similar to those in [BT08, Section 2.4.5] on how to find the optimal set of parameters that satisfies the conditions for our proof method to work, and minimizes the codelength. There we also give such minimal codelengths, for several values of c and η. Finally in Section 8.6 we prove results for asymptotically large c.

8.2 Construction

First we present the construction of the Tardos fingerprinting scheme, as in [BT08], where we use auxiliary variables d_ℓ, d_z, d_δ for the codelength ℓ, accusation offset Z and cutoff parameter δ respectively. The only difference between our construction and that of Blayer and Tassa is in the score function we use. While Blayer and Tassa used the asymmetric score function from Tardos' original scheme, we use the symbol-symmetric score function from Skoric et al.

8.2.1 The Tardos fingerprinting scheme

Let $n \geq c \geq 2$ be positive integers, and let $\epsilon_1, \epsilon_2 \in (0,1)$ be the desired upper bounds for the soundness and completeness error probabilities respectively. Let us write $k = \ln(n/\epsilon_1)$ so that $e^{-k} = \epsilon_1/n$. Let d_ℓ, d_z, d_δ be positive constants, with $d_\delta > 1$. Then the symmetric Tardos fingerprinting scheme works as follows.

1. **Initialization**

   (a) Take the codelength as $\ell = d_\ell c^2 k$.

   (b) Take the accusation offset parameter as $Z = d_z c k$.

   (c) Take the cutoff parameter as $\delta = 1/(d_\delta c)$, and compute $\delta' = \arcsin(\sqrt{\delta})$ such that $0 < \delta' < \pi/4$.

   (d) For each fingerprint position $1 \leq i \leq \ell$, select $p_i \in [\delta, 1-\delta]$ independently from the distribution defined by the following CDF $F(p)$ and PDF $f(p)$:

   $$F(p) = \frac{2\arcsin(\sqrt{p}) - 2\delta'}{\pi - 4\delta'}, \quad f(p) = \frac{1}{(\pi - 4\delta')\sqrt{p(1-p)}}. \tag{8.1}$$

   The function $f(p)$ is biased towards δ and $1-\delta$ and symmetric around $1/2$.

2. **Codeword generation**

   (a) For each position $1 \leq i \leq \ell$ and for each user $1 \leq j \leq n$, select the ith entry of the codeword of user j according to $\mathbb{P}[X_{ji} = 1] = p_i$ and $\mathbb{P}[X_{ji} = 0] = 1 - p_i$.

3. **Accusation**

   (a) For each position $1 \leq i \leq \ell$ and for each user $1 \leq j \leq n$, calculate the score S_{ji} according to:

   $$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ -\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 0, \\ +\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 0. \end{cases} \tag{8.2}$$

   (b) For each user $1 \leq j \leq n$, calculate the total accusation sum $S_j = \sum_{i=1}^{\ell} S_{ji}$. User j is accused if and only if $S_j > Z$.

Under certain conditions on the parameters d_ℓ, d_z, d_δ, which are specified below, one can prove soundness and completeness, using (a modified version of) Tardos' proof construction. Note that, since this proof method uses several non-tight bounds, it is very well possible that there exist sets of parameters that do not satisfy these conditions, but still guarantee soundness and completeness. So if the conditions are not satisfied, we can only conclude that the proof method does not work in that case.

8.2.2 Results for the asymmetric Tardos scheme

In the original Tardos scheme, and in several papers discussing the Tardos scheme, the score function is asymmetric in y_i, as only the positions with $y_i = 1$ are taken into account for the

accusations. The construction of this asymmetric Tardos scheme is the same as in Section 8.2, but with the scores from (8.2) replaced by:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{8.3}$$

Blayer and Tassa performed an extensive analysis of this scheme in [BT08], and showed that under the following assumptions, one can prove soundness and completeness for given c and η. In these Theorems, the function $h : (0, \infty) \to (\frac{1}{2}, \infty)$ is defined by $h(x) = \lambda$ if and only if $e^x = 1 + x + \lambda x^2$. The function $h^{-1} : (\frac{1}{2}, \infty) \to (0, \infty)$ denotes its inverse function, and is defined by $h^{-1}(x) = (e^x - 1 - x)/x^2$.

Theorem 8.1. *[BT08, Theorem 1.1] Let the Tardos scheme be constructed as in Section 8.2, with the asymmetric score function from* (8.3)*. Let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the following two requirements:*

$$d_\alpha \geq \frac{\sqrt{d_\delta}}{h(r)\sqrt{c}}, \tag{S1}$$

$$\frac{d_z}{d_\alpha} - \frac{r d_\ell}{d_\alpha^2} \geq 1. \tag{S2}$$

Then the scheme is ϵ_1-sound.

Theorem 8.2. *[BT08, Theorem 1.2] Let the Tardos scheme be constructed as in Section 8.2, with the asymmetric score function from* (8.3)*. Let s, g be positive constants such that d_ℓ, d_z, d_δ, s and g satisfy the following two requirements:*

$$\frac{1 - \frac{2}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g, \tag{C1}$$

$$g d_\ell - d_z \geq \eta \sqrt{\frac{d_\delta}{s^2 c}}. \tag{C2}$$

Then the scheme is ϵ_2-complete.

Tardos' original choice of parameters was the following, which allowed him to prove his scheme is ϵ_1-sound and ϵ_2-complete for all $c \geq 2$ and $\eta \leq \sqrt{c}/4$ [Tar03, Theorems 1 and 2]:

$$d_\ell = 100, \quad d_z = 20, \quad d_\delta = 300, \quad d_\alpha = 10, \quad r = 1, \quad s = 1, \quad g = \frac{1}{4}. \tag{8.4}$$

Blayer and Tassa proved that to achieve ϵ_1-soundness and ϵ_2-completeness for all $c \geq 2$ and $\eta \leq 1$, the following choice of parameters is also provably secure [BT08, Section 2.4]:

$$d_\ell = 85, \quad d_z = 15, \quad d_\delta = 40, \quad d_\alpha = 8, \quad r = 0.611, \quad s = 0.757, \quad g = 0.2461. \tag{8.5}$$

In [SVCT06, Corollary 1], Skoric et al. showed that the following choice of parameters suffices to prove soundness and completeness for asymptotically large c:

$$d_\ell \to 4\pi^2, \quad d_z \to 4\pi, \quad d_\delta \to \infty, \quad d_\alpha \to 2\pi, \quad r = 1, \quad s = h(1), \quad g \to \frac{1}{\pi}. \tag{8.6}$$

According to the Central Limit Theorem, the scores of innocent users and the total score of the coalition converge to certain normal distributions. Under the assumption that the scores behave exactly like these normal distributions, Skoric et al. showed in [SVCT06, Corollary 3]

that the following choice of parameters is then sufficient and necessary to prove soundness and completeness:

$$d_\ell \to 2\pi^2, \quad d_z \to 2\pi, \quad d_\delta \to \infty. \tag{8.7}$$

Applying the analysis from Section 8.6 to the asymmetric Tardos scheme, we can prove that the following choice of parameters is provably sufficient for large c:[1]

$$d_\ell \to 2\pi^2, \quad d_z \to 2\pi, \quad d_\delta \to \infty, \quad d_\alpha \to \pi, \quad r \to \frac{1}{2}, \quad s \to \infty, \quad g \to \frac{1}{\pi}. \tag{8.8}$$

So with Blayer and Tassa's proof construction, we obtain a 2 times shorter asymptotic codelength compared to the shortest provable codelength of Skoric et al. for the asymmetric Tardos scheme, and we achieve the asymptotic optimal codelength for the asymmetric Tardos scheme which Skoric et al. only achieved when they added the assumption that scores behave like normal distributions.

8.2.3 Results for the symmetric Tardos scheme

We will prove in Sections 8.3 and 8.4 that with the following assumptions on the parameters, we can also prove soundness and completeness for the symmetric Tardos scheme.

Theorem 8.3. *Let the Tardos scheme be constructed as in Subsection 8.2.1, and let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the requirements from* (S1) *and* (S2)*. Then the scheme is ϵ_1-sound.*

Theorem 8.4. *Let the Tardos scheme be constructed as in Subsection 8.2, and let s, g be positive constants, such that d_ℓ, d_z, d_δ, s and g satisfy* (C2) *and the following requirement:*

$$\frac{2 - \frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g. \tag{C1'}$$

Then the scheme is ϵ_2-complete.

Using the above results, in Section 8.5 we will prove ϵ_1-soundness and ϵ_2-completeness for all $c \geq 2$ and $\eta \leq 1$ for the following set of parameters:

$$d_\ell = 23.79, \quad d_z = 8.06, \quad d_\delta = 28.31, \quad d_\alpha = 4.58, \quad r = 0.67, \quad s = 1.07, \quad g = 0.49. \tag{8.9}$$

This improves upon the constants from Blayer and Tassa by a factor more than 3.5, and it improves upon the original Tardos scheme by a factor more than 4. Furthermore, for bigger c and smaller η the values of d_ℓ further decrease, easily leading to a factor 10 improvement over the original Tardos scheme.

Skoric et al. showed that for asymptotically large c, the following set of parameters is sufficient for proving soundness and completeness in the symmetric Tardos scheme [SKC08, Corollary 1]:

$$d_\ell \to \pi^2, \quad d_z \to 2\pi, \quad d_\delta \to \infty, \quad d_\alpha \to \pi, \quad r = 1, \quad s = h(1), \quad g \to \frac{2}{\pi}. \tag{8.10}$$

With the added assumption that the scores of innocent users and the joint score of guilty users are normally distributed, Skoric et al. also showed that the following set of parameters is sufficient for soundness and completeness, for asymptotically large c [SKC08, Corollary 2]:

$$d_\ell \to \frac{\pi^2}{2}, \quad d_z \to \pi, \quad d_\delta \to \infty. \tag{8.11}$$

[1] These results can be obtained by applying the analysis from Section 8.6 to Blayer and Tassa's original analysis for the asymmetric Tardos scheme. The main difference is that then one needs $g = \frac{1}{\pi} + o(1)$ instead of $g = \frac{2}{\pi} + o(1)$, which causes an extra factor 4 for d_ℓ and extra factors 2 for d_z and d_α.

Since by the Central Limit Theorem these scores will also converge to normal distributions, this shows that the asymptotic optimal codelength for the symmetric Tardos scheme is $\ell = (\frac{\pi^2}{2}+o(1))c^2\ln(n/\epsilon_1)$. We show in Section 8.6 that for asymptotically large c, we can actually prove soundness and completeness with this asymptotic codelength, without any added assumptions. In the asymptotic case, our construction gives the following parameters:

$$d_\ell \to \frac{\pi^2}{2}, \quad d_z \to \pi, \quad d_\delta \to \infty, \quad d_\alpha \to \frac{\pi}{2}, \quad r \to \frac{1}{2}, \quad s \to \infty, \quad g \to \frac{2}{\pi}. \tag{8.12}$$

Similar to the asymmetric case, we thus get a factor 2 improvement over Skoric et al.'s best provable asymptotic codelength, and we achieve the asymptotic optimal codelength which Skoric et al. only proved with the added assumption that the scores behave like normal distributions. This also improves upon results from Nuida et al. in [NFH+09], who showed that with certain discrete distribution functions F, one can prove secureness for $\ell \approx 5.35c^2\ln(n/\epsilon_1)$ for large c. With our construction, we show a codelength of $\ell \approx 4.93c^2\ln(n/\epsilon_1)$ is provably secure for large c.

8.2.4 Integral codelengths

One detail we have not taken care of and which is often "swept under the carpet" in other literature, is that the codelength ℓ by definition has to be integral. In the construction of the Tardos scheme however, we said we take $\ell = d_\ell c^2\ln(n/\epsilon_1)$, while $\ln(n/\epsilon_1)$ and d_ℓ may not be integral. To solve the problem of non-integral codelengths, Tardos rounded up $\ln(n/\epsilon_1)$ and took $d_\ell = 100$ in his original scheme. Blayer and Tassa also rounded up $\ln(n/\epsilon_1)$ and took $d_\ell = 85$, presumably also to guarantee that ℓ is integral[2]. However, rounding up d_ℓ and $\ln(n/\epsilon_1)$ could drastically increase the codelength. For example, suppose $n = 10^6$, $\epsilon_1 = \epsilon_2 = 0.01$, and $c = 25$. Then $\eta = 0.25$ and $\ln(n/\epsilon_1) \approx 18.42$, and numerical optimizations give $d_\ell \approx 8.18$. Without rounding we would get a codelength of $\ell \approx 94155$, while with rounding we get $\ell' = 106875$. So then the codelength ℓ' is more than 13.5% higher than ℓ, only because we rounded up both $\ln(n/\epsilon_1)$ and d_ℓ.

Instead of rounding up inbetween, rounding up the entire codelength to $\ell' = \lceil d_\ell c^2\ln(n/\epsilon_1)\rceil$ makes more sense. The codelength is then increased by less than 1 symbol, so we hardly notice the difference in the codelength. However, the proofs we give in Section 8.3 and 8.4 are based on $\ell = d_\ell c^2\ln(n/\epsilon_1)$, which corresponds to using $d_\ell = \ell/(c^2\ln(n/\epsilon_1))$. If we take $\ell' = \lceil\ell\rceil$, then we get $d'_\ell = \lceil\ell\rceil/(c^2\ln(n/\epsilon_1)) > d_\ell$ (for $\ell \notin \mathbb{N}$), so that with the same parameters Z and δ we may not be able to prove security anymore. In particular, equation (S2) might not be satisfied if d_ℓ is increased, since (S2) implies that $4rd_\ell \le d_z^2$. Increasing the left hand side may violate this bound, if we do not also increase d_z.

The following Theorem takes care of this minor problem, by showing that if we can find a solution to (S1), (S2), (C1'), (C2) with a fractional codelength ℓ, then we can also find a solution to these inequalities with the integral codelength $\lceil\ell\rceil$. In particular, we show which scheme parameters ℓ, Z and δ one could take to achieve this result.

Theorem 8.5. *Let the Tardos scheme be constructed as in Section 8.2, and let $(d_\ell, d_z, d_\delta, d_\alpha, r, s, g)$ be a septuple satisfying conditions* (S1), (S2), (C1'), (C2) *giving scheme parameters $\ell_0 = d_\ell c^2\ln(n/\epsilon_1), Z_0 = d_z c\ln(n/\epsilon_1)$ and $\delta_0 = 1/(d_\delta c)$. Then the Tardos scheme from Subsection 8.2 with parameters*

$$\ell = \lceil\ell_0\rceil, \quad Z = Z_0 + \frac{g}{c}\left(\lceil\ell_0\rceil - \ell_0\right), \quad \delta = \delta_0 \tag{8.13}$$

is ϵ_1-sound and ϵ_2-complete.

[2]Numerical optimizations show that even a parameter set with $d_\ell \approx 81.25$ exists that satisfies all requirements of Blayer and Tassa.

Proof. Let us write $\omega = d_\ell(\lceil \ell_0 \rceil - \ell_0)/\ell_0$. We prove that if the equations hold for $(d_\ell, d_z, d_\delta, d_\alpha, r, s, g)$, then they also hold for $(d'_\ell, d'_z, d_\delta, d'_\alpha, r, s, g)$, where $d'_\ell = d_\ell + \omega, d'_z = d_z + g\omega, d'_\alpha = (d'_z + \sqrt{(d'_z)^2 - 4rd'_\ell})/2$. Since for this set of parameters we get ℓ, Z and δ as in (8.13), the result then follows.

First note that since d_δ, s and g did not change, both sides of inequality (C1') remain the same and this inequality is still satisfied. For inequality (C2), note that both sides also remained the same, since $gd'_\ell - d'_z = g(d_\ell + \omega) - (d_z + g\omega) = gd_\ell - d_z$. For (S2), we rewrite this inequality as a quadratic inequality in d_α:

$$d_\alpha^2 + (-d_z)d_\alpha + rd_\ell \leq 0. \tag{8.14}$$

This inequality is satisfied if d_α lies between the two roots of $d_\alpha^2 + (-d_z)d_\alpha + rd_\ell = 0$, which therefore must exist. These roots exist if and only if $(d'_z)^2 - 4rd'_\ell \geq 0$. Since we know that $d_z^2 - 4rd_\ell \geq 0$ the inequality follows if

$$(d'_z)^2 - 4rd'_\ell = (d_z^2 - 4rd_\ell) + (2gd_z + g^2\omega^2 - 4r) \geq d_z^2 - 4rd_\ell \geq 0. \tag{8.15}$$

From (S2) and (C2) we know that $g(d_z^2) \geq g(4rd_\ell) \geq 4rd_z$, i.e. $gd_z \geq 4r$. So it follows that $2gd_z + g^2\omega^2 \geq 4r$, which proves the second inequality. The third inequality then follows from (S2).

Finally for (S1), we prove that $d'_\alpha \geq d_\alpha$, while the right hand side remains the same, so that this inequality is still satisfied. Note that d_α is also at most the rightmost root to (8.14), so $d'_\alpha - d_\alpha$ is bounded by

$$d'_\alpha - d_\alpha \geq \frac{d'_z + \sqrt{(d'_z)^2 - 4rd'_\ell}}{2} - \frac{d_z + \sqrt{d_z^2 - 4rd_\ell}}{2} \geq \frac{g\omega}{2} \geq 0. \tag{8.16}$$

Here the second inequality follows from earlier calculations that $(d'_z)^2 - 4rd'_\ell \geq d_z^2 - 4rd_\ell$. So this choice of d'_α is at least as high as d_α, so inequality (S1) is satisfied. This completes the proof. □

8.3 Soundness

Here we will prove Theorem 8.3, i.e. prove the soundness property, under the assumptions (S1) and (S2). We will closely follow the proof of soundness of Blayer and Tassa of [BT08, Theorem 1.1]. We will first prove an upperbound on $\mathbb{E}\left[e^{\alpha S_j}\right]$, with $\alpha = 1/(d_\alpha c)$, and then use this result to prove upper bounds on $\mathbb{P}[j \in \sigma(\vec{y})]$ for innocent users j, and $\mathbb{P}[\sigma(\rho(X)) \not\subseteq C]$.

Lemma 8.6. *Let d_α and r be positive constants, with $r > \frac{1}{2}$, such that d_δ, d_α and r satisfy Equation* (S1)*. Let j be an innocent user, and let S_j be the user's score in the Tardos scheme from Section 8.2. Let $\alpha = 1/(d_\alpha c)$. Then*

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{\alpha S_j}\right] \leq e^{-r\alpha^2\ell}. \tag{8.17}$$

Proof. First we fill in $S_j = \sum_{i=1}^{\ell} S_{ji}$ and use that S_j does not depend on $X_{j'i}$ for $j' \neq j$ to get

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{\alpha S_j}\right] = \mathbb{E}_{\vec{y},\vec{X}_j,\vec{p}}\left[\prod_{i=1}^{\ell} e^{\alpha S_{ji}}\right] = \prod_{i=1}^{\ell} \mathbb{E}_{y_i,X_{ji},p_i}\left[e^{\alpha S_{ji}}\right]. \tag{8.18}$$

Since $S_{ji} < \sqrt{1/\delta} = \sqrt{d_\delta c}$ it follows that $\alpha S_{ji} < \sqrt{d_\delta}/(d_\alpha\sqrt{c})$. From (S1) we know that $\sqrt{d_\delta}/(d_\alpha\sqrt{c}) \leq h(r)$ for our choice of r, hence $\alpha S_{ji} < h(r)$. From the definition of h we know that $e^w \leq 1 + w + rw^2$ exactly when $w \leq h(r)$. Using this with $w = \alpha S_{ji}$ we get

$$\mathbb{E}\left[e^{\alpha S_{ji}}\right] \leq \mathbb{E}\left[1 + \alpha S_{ji} + r(\alpha S_{ji})^2\right] = 1 + \alpha\mathbb{E}[S_{ji}] + r\alpha^2\mathbb{E}[S_{ji}^2]. \tag{8.19}$$

We can easily calculate $\mathbb{E}[S_{ji}]$ and $\mathbb{E}[S_{ji}^2]$, as y_i and X_{ji} are independent for innocent users j. As in [SKC08, Lemmas 2 and 3], we obtain

$$\mathbb{E}[S_{ji}] = 0, \quad \mathbb{E}[S_{ji}^2] = 1. \tag{8.20}$$

So it follows that $\mathbb{E}\left[e^{\alpha S_{ji}}\right] \leq 1 + r\alpha^2 \leq e^{r\alpha^2}$, and $\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{\alpha S_j}\right] \leq e^{r\alpha^2\ell}$, which was to be proven. □

Proof of Theorem 8.3. We prove that the probability of accusing any particular innocent user is at most ϵ_1/n. Since there are at most n innocent users, the probability of not accusing any innocent users is then at least $(1-\epsilon_1/n)^n \geq 1-\epsilon_1$, which then proves the scheme is ϵ_1-sound.

Since a user is accused if and only if his score S_j exceeds Z, we need to prove that $\mathbb{P}[S_j > Z] \leq \epsilon_1/n$ for innocent users j. First of all, we write $\alpha = 1/(d_\alpha c)$, and we use the Markov inequality and Equation (8.17) from Lemma 8.6 to obtain

$$\mathbb{P}[j \in \sigma(\vec{y})] = \mathbb{P}[S_j > Z] = \mathbb{P}\left[e^{\alpha S_j} > e^{\alpha Z}\right] \leq e^{-\alpha Z}\mathbb{E}\left[e^{\alpha S_j}\right] \leq e^{-\alpha Z + r\alpha^2\ell}. \tag{8.21}$$

Since we want to prove that $\mathbb{P}[j \in \sigma(\vec{y})] \leq \epsilon_1/n$, the proof would be complete if $e^{-\alpha Z + r\alpha^2\ell} \leq e^{-k} \leq \epsilon_1/n$, i.e. if $-\alpha Z + r\alpha^2\ell \leq -k$. Filling in $\alpha = 1/(d_\alpha c), Z = d_z ck, \ell = d_\ell c^2 k$, and dividing both sides by $-k$, we get

$$\frac{d_z}{d_\alpha} - \frac{r d_\ell}{d_\alpha^2} \geq 1. \tag{8.22}$$

This is exactly inequality (S2), which was assumed to hold. This completes the proof. □

Compared to the original proof in [BT08], this proof has barely changed. The only difference is that now the scores are counted for all positions i, instead of only those positions where $y_i = 1$. However, since in the proof in [BT08] this number of positions was bounded by ℓ, the result remains the same. This explains why we can prove ϵ_1-soundness with the symmetric score function under the same assumptions (S1), (S2) as in [BT08].

8.4 Completeness

For the proof of Theorem 8.4, we will again closely follow the proof of Blayer and Tassa of [BT08, Theorem 1.2], and make changes where necessary to incorporate the symbol-symmetric score function. We first give a Lemma to bound the expectation value of $\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right]$ with $\beta = s\sqrt{\delta}/c$ and $S = \sum_{j\in C} S_j$, and then use this Lemma to prove completeness.

Lemma 8.7. *Let s and g be positive constants such that d_δ, s and g satisfy* (C1'). *Let $\beta = s\sqrt{\delta}/c$, let C be a coalition of size c, and let $S = \sum_{j\in C} S_j$ be their total coalition score in the Tardos scheme from Section 8.2. Then*

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq e^{-g\beta\ell}. \tag{8.23}$$

Proof. First, we write the total accusation sum of all colluders together as follows:

$$S = \sum_{i=1}^{\ell}\sum_{j\in C}^{c} S_{ji} = \sum_{i=1}^{\ell} y_i\left(x_i q_i - \frac{c-x_i}{q_i}\right) + \sum_{i=1}^{\ell}(1-y_i)\left(\frac{c-x_i}{q_i} - x_i q_i\right). \tag{8.24}$$

Here x_i is the number of ones on the ith positions of all colluders, y_i is the output symbol of the pirates on position i, and we introduced the notation $q_i = \sqrt{(1-p_i)/p_i}$. Following the analysis from e.g. Blayer and Tassa, and Tardos, but using that $S_i = (1-y_i)\left(\frac{c-x_i}{q_i} - x_i q_i\right)$ for positions i where $y_i = 0$ (instead of $S_i = 0$, as with the asymmetric score function), we can bound the expectation value by

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x\right)^{\ell}, \tag{8.25}$$

where

$$M_x = \begin{cases} E_{0,x} & \text{if } x = 0, \\ E_{1,x} & \text{if } x = c, \\ \max(E_{0,x}, E_{1,x}) & \text{otherwise,} \end{cases} \tag{8.26}$$

and, for some random variable p distributed according to F,

$$E_{0,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x} e^{-\beta\left(\frac{c-x}{q} - xq\right)}\right], \tag{8.27}$$

$$E_{1,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x} e^{-\beta\left(xq - \frac{c-x}{q}\right)}\right]. \tag{8.28}$$

Now, using $\beta = s\sqrt{\delta}/c$, we bound the exponents in $E_{0,x}$ and $E_{1,x}$ as follows.

$$-s = \frac{-\beta c}{\sqrt{\delta}} \leq -\beta c q \leq -\beta\left(xq - \frac{c-x}{q}\right) \leq \frac{\beta c}{q} \leq \frac{\beta c}{\sqrt{\delta}} = s. \tag{8.29}$$

So $|\beta(xq-(c-x)/q)| \leq s$ for our choice of β. So we can use the inequality $e^w \leq 1+w+h^{-1}(s)w^2$ which holds for all $w \leq s$, with $w = \pm\beta(xq-(c-x)/q)$, to obtain

$$E_{0,x} \leq \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(1 + \beta\left(xq - \frac{c-x}{q}\right) + h^{-1}(s)\beta^2\left(xq - \frac{c-x}{q}\right)^2\right)\right], \tag{8.30}$$

$$E_{1,x} \leq \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(1 - \beta\left(xq - \frac{c-x}{q}\right) + h^{-1}(s)\beta^2\left(xq - \frac{c-x}{q}\right)^2\right)\right]. \tag{8.31}$$

Introducing more notation, this can be rewritten to

$$E_{0,x} \leq F_{0,x} + \beta F_{1,x} + h^{-1}(s)\beta^2 F_{2,x}, \tag{8.32}$$

$$E_{1,x} \leq F_{0,x} - \beta F_{1,x} + h^{-1}(s)\beta^2 F_{2,x}, \tag{8.33}$$

where

$$F_{0,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\right], \tag{8.34}$$

$$F_{1,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(xq - \frac{c-x}{q}\right)\right], \tag{8.35}$$

$$F_{2,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(xq - \frac{c-x}{q}\right)^2\right]. \tag{8.36}$$

We first calculate $F_{1,x}$ explicitly. Writing out the expectation value and using the definition of $f(p)$ from (8.1), we get

$$F_{1,x} = \frac{1}{\pi - 4\delta'}\int_{\delta}^{1-\delta} p^x(1-p)^{c-x}\left(\frac{x}{p} - \frac{c-x}{1-p}\right)dp \tag{8.37}$$

The primitive of the integrand is given by $I(p) = p^x(1-p)^{c-x}$, so we get

$$F_{1,x} = \frac{I(1-\delta) - I(\delta)}{\pi - 4\delta'} = \frac{(1-\delta)^x\delta^{c-x} - \delta^x(1-\delta)^{c-x}}{\pi - 4\delta'}. \tag{8.38}$$

For $0 < x < c$, we bound $F_{1,x}$ from above and below as

$$\frac{-\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} \leq F_{1,x} \leq \frac{(1-\delta)^x\delta^{c-x}}{\pi - 4\delta'}. \tag{8.39}$$

Using these bounds for M_x, with $0 < x < c$, we get

$$M_x \leq F_{0,x} + \beta\frac{\max(\delta^x(1-\delta)^{c-x}, (1-\delta)^x\delta^{c-x})}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,x}. \tag{8.40}$$

Since $\delta < 1 - \delta$, the maximum of the two terms is the first term when $0 < x \leq c/2$, and it is the second term when $c/2 < x < c$. Note that this bound is different from the one of Blayer and Tassa, since in their analysis they do not have this maximum over two terms, but just the first of these two terms. We cannot prove the same upper bound as Blayer and Tassa, and therefore our bound for $M_x, 0 < x < c$, is slightly weaker than Blayer and Tassa's.

For the positions where the marking assumption applies, i.e. $x = 0$ and $x = c$, we do not use the bounds on $F_{1,x}$, but use the exact formula from (8.38) to obtain

$$M_0 \leq F_{0,0} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,0}, \tag{8.41}$$

$$M_c \leq F_{0,c} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,c}. \tag{8.42}$$

The value of M_c is the same as that of Blayer and Tassa, but whereas Blayer and Tassa had $M_0 = F_0$, we get a lower upper bound on M_0. This is essentially the reason why with the symmetric score function we get shorter codelengths than Blayer and Tassa.

Substituting the bounds on M_x in the summation over M_x from (8.25) gives us

$$\sum_{x=0}^{c}\binom{c}{x}M_x \leq M_0 + M_c + \sum_{x=1}^{c-1}\binom{c}{x}M_x \tag{8.43}$$

$$\leq \left(F_{0,0} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,0}\right) \tag{8.44}$$

$$+ \left(F_{0,c} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,c}\right) \tag{8.45}$$

$$+ \sum_{x=1}^{\lfloor c/2\rfloor}\binom{c}{x}\left(F_{0,x} + \beta\frac{\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,x}\right) \tag{8.46}$$

$$+ \sum_{x=\lfloor c/2\rfloor+1}^{c-1}\binom{c}{x}\left(F_{0,x} + \beta\frac{(1-\delta)^x\delta^{c-x}}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,x}\right). \tag{8.47}$$

Gathering all terms with $F_{0,x}$ and $F_{2,x}$, and using the substitution $x' = c - x$ for the summation over $\lceil c/2\rceil - 1$ terms, we get

$$\sum_{x=0}^{c}\binom{c}{x}M_x \leq \sum_{x=0}^{c}\binom{c}{x}F_{0,x} - \beta\frac{2(1-\delta)^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2\sum_{x=0}^{c}\binom{c}{x}F_{2,x} \tag{8.48}$$

$$+ \frac{\beta}{\pi - 4\delta'}\left(\delta^c + \sum_{x=1}^{\lfloor c/2\rfloor}\binom{c}{x}\delta^x(1-\delta)^{c-x}\right) \tag{8.49}$$

$$+ \frac{\beta}{\pi - 4\delta'}\left(\delta^c + \sum_{x'=1}^{\lceil c/2\rceil-1}\binom{c}{x'}\delta^{x'}(1-\delta)^{c-x'}\right). \tag{8.50}$$

For the summation over $F_{2,x}$, let us define a sequence of random variables $\{T_i\}_{i=1}^{c}$ according to $T_i = q$ with probability p and $T_i = -1/q$ with probability $1-p$. Similar to the inequalities from (8.20), we get that $\mathbb{E}_p[T_i] = 0$ and $\mathbb{E}_p[T_i^2] = 1$. Also, since T_i and T_j are independent for $i \neq j$, we have that $\mathbb{E}_p[T_i T_j] = 0$ for $i \neq j$. Therefore we can write

$$\mathbb{E}_p\left[\left(\sum_{i=1}^{c} T_i\right)^2\right] = \sum_{i=1}^{c} \mathbb{E}_p\left[T_i^2\right] + \sum_{i\neq j} \mathbb{E}_p\left[T_i T_j\right] = c. \tag{8.51}$$

But writing out the definition of the expected value, we see that the left hand side is actually the same as the summation over $F_{2,x}$, so that we get

$$\mathbb{E}_p\left[\left(\sum_{i=1}^{c} T_i\right)^2\right] = \sum_{x=0}^{c} \binom{c}{x} p^x (1-p)^{c-x} \left(xq - \frac{c-x}{q}\right)^2 = \sum_{x=0}^{c} \binom{c}{x} F_{2,x} = c. \tag{8.52}$$

Also we trivially have that

$$\sum_{x=0}^{c} \binom{c}{x} F_{0,x} = \sum_{x=0}^{c} \binom{c}{x} \mathbb{E}_p\left[p^x (1-p)^{c-x}\right] = \mathbb{E}_p\left[\sum_{x=0}^{c} \binom{c}{x} p^x (1-p)^{c-x}\right] = 1. \tag{8.53}$$

For the summation over $\lfloor c/2 \rfloor$ terms we use the following upper bound, which then also holds for the summation over $\lceil c/2 \rceil - 1$ terms:

$$\delta^c + \sum_{x=1}^{\lfloor c/2 \rfloor} \binom{c}{x} \delta^x (1-\delta)^{c-x} \leq \sum_{x=1}^{c} \binom{c}{x} \delta^x (1-\delta)^{c-x} = 1 - (1-\delta)^c \leq \delta c. \tag{8.54}$$

Note that this first inequality is quite sharp. In most cases $\delta \ll 1-\delta$, so that the summation is dominated by the terms with low values of x. Adding the terms with $\lfloor c/2 \rfloor < x < c$ (i.e. terms with high powers of δ) to the summation has an almost negligible effect on the value of the summation.

Now applying the previous results to (8.50), and using $(1-\delta)^c \geq 1 - \delta c$, which holds for all c, we get

$$\sum_{x=0}^{c} \binom{c}{x} M_x \leq 1 - \beta \frac{2 - 4c\delta}{\pi - 4\delta'} + h^{-1}(s)\beta^2 c. \tag{8.55}$$

We want to prove that, for some $g > 0$,

$$\sum_{x=0}^{c} \binom{c}{x} M_x \leq 1 - \beta \frac{2 - 4c\delta}{\pi - 4\delta'} + h^{-1}(s)\beta^2 c \leq 1 - g\beta \leq e^{-g\beta}. \tag{8.56}$$

Filling in $\beta = s\sqrt{\delta}/c$ and $\delta = 1/(d_\delta c)$ and writing out the second inequality, this leads to the requirement that

$$\frac{2 - \frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g. \tag{8.57}$$

This is exactly inequality (C1'), which is assumed to hold. So combining the results from Equations (8.56) and (8.50) gives us that

$$\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x\right)^{\ell} \leq e^{-g\beta\ell}. \tag{8.58}$$

This completes the proof. □

Proof of Theorem 8.4. We will prove that for a coalition of size c, with probability at least $1-\epsilon_2$ the algorithm will accuse at least one of the colluders. Since a coalition of size c can simulate a smaller coalition by disregarding some of its users, this then proves that for any coalition of size at most c, the scheme will accuse at least one of the colluders with probability at least $1-\epsilon_2$. Note that if no colluders are accused, then the score of each colluder is below Z. Hence if the total coalition score S exceeds cZ, then at least one of the pirates is accused. So to prove ϵ_2-soundness, it suffices to prove that $\mathbb{P}[S < cZ] \leq \epsilon_2$.

We first use the Markov inequality and Lemma 8.7 with $\beta = s\sqrt{\delta}/c > 0$ to get

$$\mathbb{P}[\sigma(\vec{y}) \cap C = \emptyset] \leq \mathbb{P}[S < cZ] = \mathbb{P}\left[e^{-\beta S} > e^{-\beta cZ}\right] \leq e^{\beta cZ}\mathbb{E}_{\vec{y},X,\vec{p}}\left[e^{-\beta S}\right] \leq e^{\beta cZ - g\beta\ell}. \tag{8.59}$$

Since we want to prove that $\mathbb{P}[S < cZ] \leq e^{-\eta k} \leq (\epsilon_1/n)^\eta = \epsilon_2$, the proof would be complete if $\beta cZ - g\beta\ell \leq -\eta k$. Filling in $\beta = s\sqrt{\delta}/c, \ell = d_\ell c^2 k, Z = d_z ck, \delta = 1/(d_\delta c)$ and writing out both sides, we get

$$gd_\ell - d_z \geq \eta\sqrt{\frac{d_\delta}{s^2 c}}. \tag{8.60}$$

This is exactly inequality (C2), which was assumed to hold. This completes the proof. □

Compared to [BT08], we see that instead of using (C1), we now need that inequality (C1') holds. Comparing these two inequalities, we see that a term $\frac{1}{\pi}$ has changed to a $\frac{2}{\pi}$, and a term $\frac{2}{d_\delta \pi}$ has changed to a $\frac{4}{d_\delta \pi}$. The most important change is the $\frac{1}{\pi}$ changing to a $\frac{2}{\pi}$, since that term is the most dominant factor (and the only positive term) on the left hand side of (C1'). By increasing this by a factor 2, we get that $g \leq \frac{2}{\pi}$ instead of $g \leq \frac{1}{\pi}$. Especially for large c, this will play an important role, and it will basically be the reason why the required codelength can then be reduced by a factor 4, compared to Blayer and Tassa's analysis for the asymmetric scheme.

While the other change (the $\frac{2}{d_\delta \pi}$ changing to $\frac{4}{d_\delta \pi}$) does not have a big impact on the optimal choice of parameters for large c, this change does influence the required codelength for smaller c. Because of this change, we now subtract more from the left hand side of (C1'), so that the value of g is bounded sharper from above. This means that for finite values of c, we cannot reduce the codelength of Blayer and Tassa by a factor 4, but only by a factor slightly less than 4.

Finally, after using (C1') in the proof above, the analysis remained the same as in [BT08]. So under the same assumption (C2) as in [BT08], we could also complete the proof for the symmetric Tardos scheme.

8.5 Optimization

Similar to the analysis done by Blayer and Tassa in [BT08, Section 2.4], we also investigate the optimal choice of parameters such that all requirements are satisfied, and d_ℓ is minimized. As only one of the inequalities has changed, and it changed only on two positions, the formulas for the optimal values of $d_\delta, d_\alpha, d_z, d_\ell$ in the following Theorem are almost the same as in [BT08, Section 2.4.5].

Theorem 8.8. *Let η, c be given, and let r, s, g be fixed, satisfying $r \in (\frac{1}{2}, \infty), s \in (0, \infty), g \in (0, \frac{2}{\pi})$. Then the optimal choice of $d_\delta, d_\alpha, d_z, d_\ell$, minimizing d_ℓ and satisfying conditions*

(S1),(S2),(C1'),(C2), *is given by:*

$$\hat{d}_\delta = \left(\frac{1}{\frac{4}{\pi} - 2g} \left(\sqrt{\frac{(h^{-1}(s)s)^2}{c} + \frac{16}{\pi}\left(\frac{2}{\pi} - g\right)} + \frac{h^{-1}(s)s}{\sqrt{c}} \right) \right)^2, \tag{O1}$$

$$\hat{d}_\alpha = \max\left(\frac{\sqrt{\hat{d}_\delta}}{h(r)\sqrt{c}}, \frac{r}{g} + \sqrt{\left(\frac{r}{g}\right)^2 + \frac{r}{g}\eta\sqrt{\frac{\hat{d}_\delta}{s^2 c}}} \right), \tag{O2}$$

$$\hat{d}_z = \frac{g\hat{d}_\alpha^2 + r\eta\sqrt{\frac{\hat{d}_\delta}{s^2 c}}}{g\hat{d}_\alpha - r}, \tag{O3}$$

$$\hat{d}_\ell = \frac{\eta\sqrt{\frac{\hat{d}_\delta}{s^2 c}} + \hat{d}_z}{g}. \tag{O4}$$

So to find the optimal septuple $(\hat{r}, \hat{s}, \hat{g}, \hat{d}_\delta, \hat{d}_\alpha, \hat{d}_z, \hat{d}_\ell)$ for given c, η, satisfying all requirements and minimizing $\hat{d}_\ell$, one only has to find the triple (r, s, g) with $r \in (\frac{1}{2}, \infty)$, $s \in (0, \infty)$ and $g \in (0, \frac{2}{\pi})$ that minimizes the right hand side of (O4).

Example An optimal solution to (S1),(S2),(C1'),(C2) for $c \geq 2$ and $\eta = 1$, minimizing d_ℓ, is given by

$$d_\ell = 23.79, \quad d_z = 8.06, \quad d_\delta = 28.31, \quad d_\alpha = 4.58, \quad r = 0.67, \quad s = 1.07, \quad g = 0.49. \tag{8.61}$$

This means that with these constants, we can prove soundness and completeness for all $c \geq 2$ and $\eta \leq 1$, with a codelength of $\ell = 23.79c^2 \ln(n/\epsilon_1)$. Compared to the original Tardos scheme, which had a codelength of $\ell = 100c^2 \lceil \ln(n/\epsilon_1) \rceil$, this gives an improvement of a factor more than 4. Furthermore we can prove that this scheme is ϵ_1-sound and ϵ_2-complete for any value of $c \geq 2$ and $\eta \leq 1$, while Tardos' original proof only works for $c \geq 2$ and $\eta \leq \sqrt{c}/4$.

Example In practice, one usually has $\eta \ll 1$ instead of $\eta = 1$. For example, it could be that $\epsilon_2 = 1/2$ is sufficient, while $\epsilon_1 = 10^{-3}$ is desired and there are $n = 10^6$ users, so that $\eta \approx 0.033$. Then the optimizations give us $d_\ell \approx 10.89$ for $c = 2$. So with this larger value of ϵ_2, a codelength of $\ell = 10.89c^2 \ln(n/\epsilon_1)$ is sufficient to prove the soundness and completeness properties for any $c \geq 2$. This is then already a factor more than 9 improvement compared to the original Tardos scheme.

If we let c increase in inequalities (O1),(O2),(O3),(O4), i.e. if we only want provable security for $c \geq c_0$ for some $c_0 > 2$, then one can easily see that the inequalities become weaker and an even shorter codelength can be achieved. Figure 8.1 shows the optimal values of d_ℓ against different values of c, for several values of η. One can see that for large c, a codelength of $\ell < 6c^2 \ln(n/\epsilon_1)$ can be sufficient. In the next section, we will see that for large c, the optimal values of d_ℓ will converge to $\frac{\pi^2}{2} \approx 4.93$.

8.6 Asymptotics

We now look at the asymptotic behaviour of the scheme as c goes to infinity. When c goes to infinity, the distributions of the scores of both guilty and innocent users converge to the Normal distribution with certain parameters. In [SKC08, Section 6] Skoric et al. investigated this Gaussian approximation, and concluded that with Tardos' choice of g_0, g_1 and F, the required

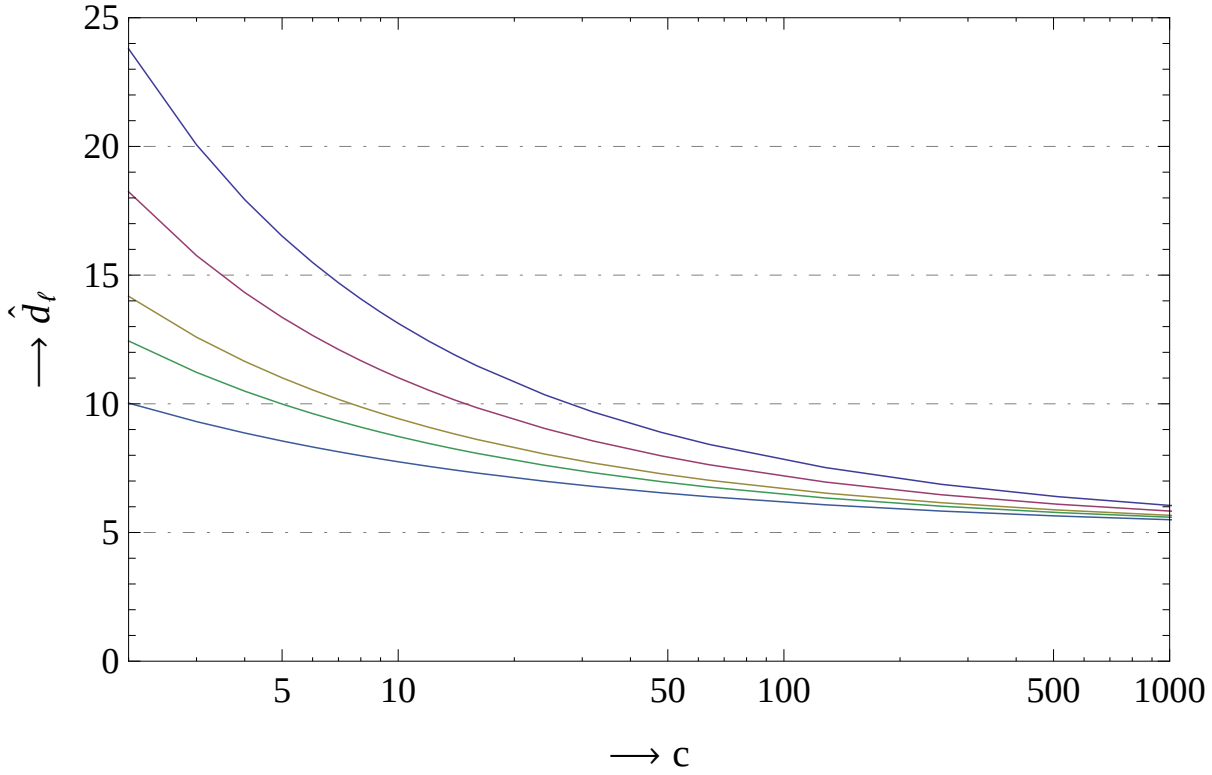

Figure 8.1: *Optimal values of d_ℓ, for several values of c between 2 and 1000. The different lines correspond to the cases $\eta = 1, 0.5, 0.2, 0.1, 0.01$ respectively, where higher values of η correspond to higher values of $\hat{d}_\ell$.*

codelength is $\ell \approx \frac{\pi^2}{2} c^2 \ln(n/\epsilon_1)$. This means that for sufficiently large c we will certainly need that $d_\ell \geq \frac{\pi^2}{2}$.

In Tardos' original paper, Tardos proved that $d_\ell = 100$ is sufficient for $c \geq 16$. This shows that either Tardos' choice of parameters was not optimal, or that the proof method is not tight. In [SKC08] the symmetric accusations were introduced, showing that even $d_\ell > \pi^2$ is sufficient for proving soundness and completeness, for sufficiently large c. In [BT08] the analysis of the scheme, which was already tightened in [SKC08], was further tightened, but no symmetric accusations were used. Applying asymptotics to their scheme shows that using their analysis, $d_\ell > 2\pi^2$ is sufficient for proving security.

Here we will show that by combining the symmetric accusations from Skoric et al. with the tighter analysis from Blayer and Tassa, as we did above, we can prove security for $d_\ell > \frac{\pi^2}{2}$. This means that the gap of a factor 2 between provability and reality, as in [SKC08], has now been closed. This is also why we refer to our scheme as the optimal Tardos scheme, as for $c \to \infty$ our scheme achieves the theoretical optimal codelength.

Theorem 8.9. *For $c \gg 1$ the above construction gives an ϵ_1-sound and ϵ_2-complete scheme with parameters*

$$\ell \approx \frac{\pi^2}{2} c^2 \lceil \ln(n/\epsilon_1) \rceil \qquad Z \approx \pi c \lceil \ln(n/\epsilon_1) \rceil \tag{8.62}$$

Proof. In our scheme we will take optimal values of $d_\ell, d_z, d_\delta, d_\alpha, r, s, g$ such that all requirements are met and d_ℓ is minimized. Hence showing that *some* parameters $d_\ell, d_z, d_\delta, d_\alpha, r, s, g$ exist, which meet all requirements and have $d_\ell \downarrow \frac{\pi^2}{2}$ as $c \to \infty$ is sufficient for proving the Theorem.

For the second requirement, note that we can write it as a quadratic inequality in d_α, as

$$d_\alpha^2 + (-d_z)d_\alpha + rd_\ell \leq 0 \tag{8.63}$$

The constant in front of d_α is positive, while the function has to be negative. So this requirement is met if and only if d_α lies between its two roots, which therefore must exist.

$$d_\alpha \in [d_{\alpha,-}, d_{\alpha,+}] = \left[\frac{d_z}{2} - \frac{1}{2}\sqrt{d_z^2 - 4rd_\ell}, \frac{d_z}{2} + \frac{1}{2}\sqrt{d_z^2 - 4rd_\ell}\right] \tag{8.64}$$

Hence taking $d_\alpha = \frac{d_z}{2}$ always satisfies this equation. The only remaining requirement is then that the quadratic equation in d_α in fact has a real-valued solution. So we need that the term inside the square root is non-negative, i.e.

$$d_z^2 \geq 4rd_\ell \tag{8.65}$$

For the first requirement we then see that with $d_\delta = \mathcal{O}(\ln(c))$ and $r = \frac{1}{2} + \mathcal{O}(1/\ln(c))$, the right hand side converges to 0 as $c \to \infty$. Since the left hand side, $d_\alpha = \frac{d_z}{2}$, is positive (our d_z will converge to $\pi > 0$), this requirement will certainly be satisfied for sufficiently large c.

For the third requirement, note that the terms $\frac{4}{d_\delta \pi}$ and $\frac{h^{-1}(s)s}{\sqrt{d_\delta c}}$ both converge to 0 as c goes to infinity. This means that for sufficiently large c, the inequality will converge to

$$\frac{2}{\pi} \geq g \tag{8.66}$$

For the fourth requirement, again note that the term on the right hand side disappears as c goes to infinity. So this inequality converges to

$$gd_\ell - d_z \geq 0 \tag{8.67}$$

Taking $g \approx 2/\pi$ and solving these equations gives us

$$d_z \geq 2r\pi \tag{8.68}$$
$$d_\ell \geq r\pi^2 \tag{8.69}$$

With $r = \frac{1}{2} + \mathcal{O}(1/\ln(c)) \to \frac{1}{2}$ as $c \to \infty$, we thus get

$$d_z > \pi \tag{8.70}$$
$$d_\ell > \frac{\pi^2}{2} \tag{8.71}$$

By taking c sufficiently large, one can thus get d_ℓ arbitrarily close to $\frac{\pi^2}{2}$, as was to be shown. □

Note that near the end of the above proof, we had the two equations

$$d_z \geq 2r\pi \tag{8.72}$$
$$d_\ell \geq r\pi^2 \tag{8.73}$$

Here we used that r can be taken in the neighborhood of $\frac{1}{2}$ to get the final result, $d_\ell > \frac{\pi^2}{2}$. In [SKC08] however, no such variable r was used, as it was simply fixed at 1. Taking $r = 1$ in these equations indeed gives

$$d_z \geq 2\pi \tag{8.74}$$
$$d_\ell \geq \pi^2 \tag{8.75}$$

as was the result in [SKC08, Section 5.2]. This thus shows where the proof by Skoric et al. lost the factor 2 in the asymptotic case; if they had taken r as a parameter in their analysis, they would have gotten the same asymptotic results as we did above.

An immediate consequence of Theorem 8.9 is the following result, which shows that for large c we will achieve codelengths of $\ell \approx 4.93c^2 \ln(n/\epsilon_1)$, i.e. codelengths that are about 4.93% of Tardos' original codelengths.

Corollary 8.10. *For $c \to \infty$ the above construction gives an ϵ_1-sound and ϵ_2-complete scheme with parameters*

$$\ell \to \frac{\pi^2}{2} c^2 \ln(n/\epsilon_1), \quad Z \to \pi c \ln(n/\epsilon_1), \quad \delta \to 0. \tag{8.76}$$

This proves that our symmetric Tardos construction is asymptotically optimal, since for large c we achieve the optimal codelength of $\ell = (\pi^2 + o(1))c^2 \ln(n/\epsilon_1)$.

Remark In the proof of Theorem 8.9, we use that r can be taken in the neighborhood of $\frac{1}{2}$ to get the final result, $d_\ell = \frac{\pi^2}{2} + O(c^{-1/3+\gamma})$. In [SKC08] however, no such variable r was used, as it was simply fixed at 1. Taking $r = 1$ in the proof of Theorem 8.9 gives us:

$$\hat{d}_\delta = O(c^{1/3}), \quad \hat{d}_\alpha = \pi + O(c^{-1/3}), \quad \hat{d}_z = 2\pi + O(c^{-1/3}), \quad \hat{d}_\ell = \pi^2 + O(c^{-1/3}). \tag{8.77}$$

This thus gives a codelength of $\ell \to \pi^2 c^2 \ln(n/\epsilon_1)$, which was also the result in [SKC08, Section 5.2]. This shows where Skoric et al. lost the factor 2 in the asymptotic case; if they had taken r as a parameter in their analysis and had taken it close to $\frac{1}{2}$ in the asymptotic case, then they would have obtained the same asymptotic results as we did above.

8.7 Summary

In this chapter we improved upon the results from literature, by proving a codelength of $\ell = (\frac{\pi^2}{2} + o(1))c^2 \ln(n/\epsilon_1)$ is sufficient for secureness for large c. This improves upon the main result of Skoric et al. in [SKC08] by a factor 2, it improves upon a result of Skoric et al. by proving the same codelength without needing the requirement that the distribution behaves like a normal distribution, and it improves upon the asymptotic codelength from Nuida et al. in [NFH+09] who prove $\ell \approx 5.35c^2 \ln(n/\epsilon_1)$ for large c. We also showed how we improved upon the results from Skoric et al., by indicating where we gained the factor 2.

Also for finite c, our codelengths are shorter than those of Blayer and Tassa, Skoric et al., and in some cases also shorter than those of Nuida et al. The improvement over the original scheme of Tardos is at least a factor 4 (for $c = 2, \eta = 1$) but can be as high as 20 (for large c).

Chapter 9

The dynamic Tardos scheme

9.1 Introduction

In this chapter we will discuss a dynamic version of the Tardos scheme. Whereas the normal Tardos scheme discussed earlier belongs in the category of probabilistic static schemes, we will show how to construct a probabilistic dynamic scheme based on the Tardos scheme, and why it has advantages over the original Tardos scheme. Note that unlike with other dynamic schemes discussed in Part I, here we do need to know (an upper bound on) c in advance. In the next chapter we show how we can remove this dependence on c, giving a fully dynamic Tardos scheme.

Since our dynamic Tardos scheme makes use of analysis from the static Tardos scheme, we will make use of the improved Tardos scheme from Chapter 8 as a building block. Not only does it achieve low provably secure codelengths, but also does the proof of completeness there make use of a variable $\beta = \mathcal{O}(\sqrt{\delta}/c)$ instead of $\beta = \mathcal{O}(1/c)$, as was done in Tardos' extended article to prove completeness for $c \leq 15$. The reason why this is useful will appear later.

Note that while Tassa obtained a probabilistic dynamic scheme by replacing symbols in a deterministic dynamic scheme (the Fiat-Tassa scheme) with codewords from a probabilistic static scheme (the Boneh-Shaw scheme), here instead we start with a probabilistic static scheme (the Tardos scheme) and try to use the fact that we are working in a dynamic setting to get better results.

9.2 Construction

Let us start with the construction, which can be summarized in a few lines. Instead of distributing all symbols of the Tardos codewords simultaneously, we give users one symbol at a time. And instead of looking at scores only at time ℓ, we now calculate scores $S_j(t) = \sum_{i=1}^{t} S_{ji}$ at every time step. And most importantly: at any time t, we disconnect all users with scores $S_j(t) > Z$. Figure 9.1 graphically shows the difference between the construction of the old Tardos scheme, and the construction of our scheme.

This scheme differs a lot from dynamic schemes like the Fiat-Tassa scheme and the Berkman scheme, in the sense that the codewords are in fact independent of the pirate output. One would expect that to achieve better results, one should use the pirate output to optimize the symbol distribution, but here we show that using only the fact that we can disconnect users inbetween already leads to big improvements.

Since the codewords can be determined in advance, the exact implementation of the scheme

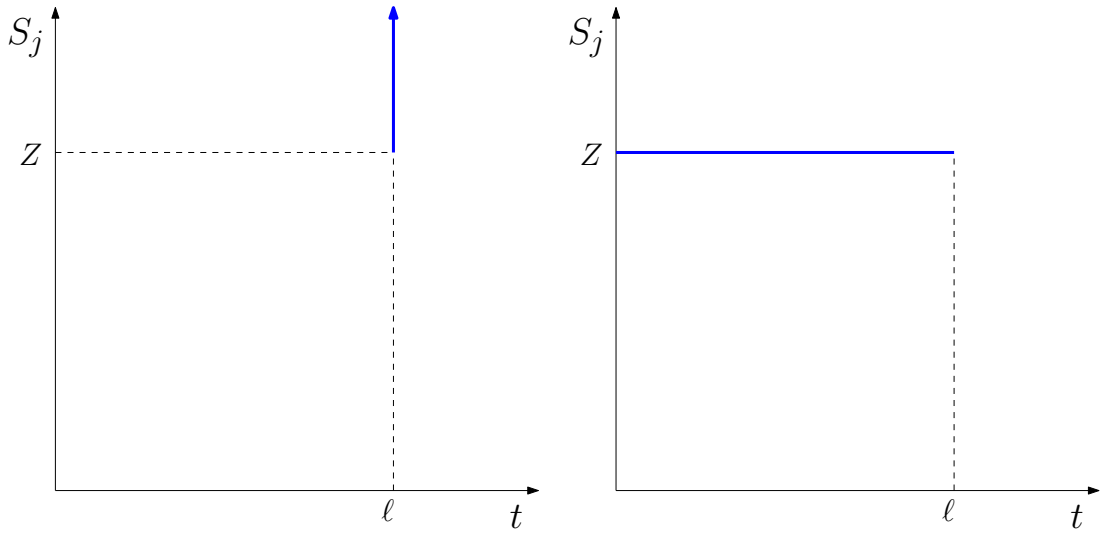

Figure 9.1: *The essential difference between the regular Tardos scheme and the dynamic Tardos scheme. While in the static Tardos scheme users are disconnected once they intersect the blue line in the left figure (i.e. if $S_j(\ell) > Z$), in the dynamic scheme users are now disconnected once they cross the blue line on the right (i.e. if $S_j(t) > Z$ for some $0 \leq t \leq \ell$).*

may differ, depending on the setting. Below we give an outline of the scheme where codewords are generated in advance, and therefore could theoretically be stored at the client side.

Let $n \geq c \geq 2$ be positive integers, and let $\epsilon_1, \epsilon_2 \in (0, 1)$ be the desired upper bounds for the soundness and completeness error probabilities respectively. Let us write $k = \ln(n/\epsilon_1)$ so that $e^{-k} = \epsilon_1/n$. Let d_ℓ, d_z, d_δ be positive constants, with $d_\delta > 1$. Then the dynamic Tardos fingerprinting scheme works as follows.

1. **Initialization**

   (a) Take the codelength as $\ell = d_\ell c^2 k$.

   (b) Take the accusation offset parameter as $Z = d_z c k$.

   (c) Take the cutoff parameter as $\delta = 1/(d_\delta c)$, and compute $\delta' = \arcsin(\sqrt{\delta})$.

   (d) For each fingerprint position $1 \leq i \leq \ell$, select $p_i \in [\delta, 1-\delta]$ independently from the distribution defined by the following CDF $F(p)$ and PDF $f(p)$:

$$F(p) = \frac{2\arcsin(\sqrt{p}) - 2\delta'}{\pi - 4\delta'}, \quad f(p) = \frac{1}{(\pi - 4\delta')\sqrt{p(1-p)}}. \tag{9.1}$$

   The function $f(p)$ is biased towards δ and $1-\delta$ and symmetric around $1/2$.

2. **Codeword generation**

   (a) For each position $1 \leq i \leq \ell$ and for each user $1 \leq j \leq n$, select the ith entry of the codeword of user j according to $\mathbb{P}[X_{ji} = 1] = p_i$ and $\mathbb{P}[X_{ji} = 0] = 1 - p_i$.

3. **Distribution/Accusation**

   (a) For each position $1 \leq i \leq \ell$ do the following:

      i. Send to each connected user j the ith symbol X_{ji}.

ii. After intercepting the pirate output y_i, calculate the score S_{ji} according to:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ -\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 0, \\ +\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 0. \end{cases} \tag{9.2}$$

(If no pirate output is detected, then the whole coalition has already been caught and disconnected, and we simply terminate.)

iii. For each active user j, update the accusation sum $S_j(i)$ according to $S_j(i) = S_j(i-1) + S_{ji}$, i.e. $S_j(i) = \sum_{k=1}^{i} S_{jk}$. If $S_j(i) > Z$, then disconnect user j immediately.

Compared to the static Tardos scheme, the main difference is that users are disconnected inbetween as well, instead of only at the end. So why does this help? Well, for innocent users we expect that things do not change a lot, compared to the static Tardos scheme. The probability of accidentally disconnecting an innocent user increases, since an innocent user could have had $S_j(t) > Z$ at some time $t < \ell$, while $S_j(\ell) < Z$. But we will show that, compared to the static Tardos scheme, this probability increases by at most a factor 2.

For guilty users however, we get an important advantage, based on the proof construction from the original Tardos scheme. There, to prove that at least one guilty user gets accused, we proved that $S < cZ$ occurs only with low probability. In all other cases, by the pigeonhole principle, at least one of the scores will be above Z, hence at least one pirate is caught. But now, since we throw out users as soon as their scores exceed Z, we know that pirates will actually never get a score higher than $Z' = Z + \max_p S_{ji}(p) < Z + 1/\sqrt{\delta}$, which is relatively close to Z. So the probability of catching *all colluders* is in fact related to $\mathbb{P}[S < cZ']$: if not all pirates are caught, then it follows that $S < cZ'$. And since $\mathbb{P}[S < cZ'] \approx \mathbb{P}[S < cZ]$ for $Z' \approx Z$, we see that the probability of *not catching all colluders* can now be bounded from above by roughly the same ϵ_2 as the one bounding the probability of *not catching any colluders* in the static scheme. So by following the Tardos analysis, we can show that the dynamic Tardos scheme will catch *all colluders* with high probability, and will not disconnect any innocent users with high probability.

For the construction, we again used auxiliary variables d_ℓ, d_z and d_δ, like we did in Chapter 8. As it turns out, we can prove soundness and (special) completeness using the following four assumptions. Note that the assumptions have barely changed compared to Chapter 8, and since asymptotically $s \leq \log(c^{1/3}) \ll \log(n/\epsilon_1)$ for $c \ll n$, the influence of the changes also disappears as $c \to \infty$.

Theorem 9.1. *Let the dynamic Tardos scheme be constructed as above. Let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the following two requirements:*

$$d_\alpha \geq \frac{\sqrt{d_\delta}}{h(r)\sqrt{c}}, \tag{S1}$$

$$\frac{d_z}{d_\alpha} - \frac{r d_\ell}{d_\alpha^2} \geq 1 + \frac{\ln(2)}{k}. \tag{S2*}$$

Then the probability of accusing some innocent users is at most ϵ_1, i.e. the scheme is ϵ_1-sound.

Theorem 9.2. *Let the dynamic Tardos scheme be constructed as above. Let s, g be positive*

constants such that d_ℓ, d_z, d_δ, s and g satisfy the following two requirements:

$$\frac{2-\frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g, \tag{C1'}$$

$$gd_\ell - d_z \geq \left(\eta + \frac{s+\ln(2)}{k}\right)\sqrt{\frac{d_\delta}{s^2 c}}. \tag{C2*}$$

Then the probability of not accusing all guilty users is at most ϵ_2, i.e. the scheme is ϵ_2-complete.

The completeness-property stated above is different from the completeness-property in the static setting. Here we require that *all* pirates are caught, instead of at least one.

9.3 Soundness

First we prove an upper bound on the probability that an innocent user gets accused. This bound relates the false positive probability in the dynamic Tardos scheme to the probability that the score at time ℓ is above Z. We then use the proof of the static Tardos scheme to get an absolute upper bound on the false positive error probability, and to prove Theorem 9.1.

Lemma 9.3. *Let $j \in U$ be an arbitrary user, and let $C \subseteq U \setminus \{j\}$ be a coalition of any size not containing j. Let ρ be some pirate strategy employed by this coalition. Then*

$$\mathbb{P}[j \in \sigma(\rho(X))] \leq 2 \cdot \mathbb{P}[S_j(\ell) > Z] \tag{9.3}$$

In other words, the probability of disconnecting an innocent user j in the dynamic scheme is at most a factor 2 bigger than the probability of disconnecting an innocent user j in the static Tardos scheme.

Proof. Let A be the event that an innocent user j gets accused in our dynamic Tardos scheme, i.e. A is the event that $S_j(t_0) \geq Z$ for some $t_0 \in \{0, \ldots, \ell\}$. Let B be the event that $S_j(\ell) > Z$. Now trivially $\mathbb{P}[A|B] = 1$. For $\mathbb{P}[B|A]$, note that the conditional gives us that there exists some time t_0 such that $S_j(t_0) = Z + \alpha_0$ for some $\alpha_0 \in [0, \sqrt{1/\delta}]$. Since the process $\{S_j(t)\}_{t=t_0}^{\infty}$ starting at time t_0 describes a random walk with zero drift, we have $\mathbb{P}[S_j(\ell) \geq S_j(t_0)] = 1/2$ and thus $\mathbb{P}[S_j(\ell) \geq Z | S_j(t_0) \geq Z] \geq 1/2$.[1] So $\mathbb{P}[B|A] \geq 1/2$. Finally we apply Bayes' Theorem, which says that for any two events A and B, $\mathbb{P}[A]\mathbb{P}[B|A] = \mathbb{P}[B]\mathbb{P}[A|B]$. Applying this to our A and B gives us $\mathbb{P}[A] \leq 2 \cdot \mathbb{P}[B]$, hence $\mathbb{P}[j \in \sigma(\rho(X))] \leq 2 \cdot \mathbb{P}[S_j(\ell) > Z]$. □

Lemma 9.4. *Let $j \in U$ be an arbitrary user, and let $C \subseteq U \setminus \{j\}$ be a coalition of any size not containing j. Let ρ be some pirate strategy employed by this coalition. Let* (S1) *and* (S2*) *be satisfied. Then*

$$\mathbb{P}[S_j(\ell) > Z] \leq \frac{\epsilon_1}{2n}. \tag{9.4}$$

Proof. First, we remark that the distribution of $S_j(\ell)$ is the same as the distribution of S_j in the static Tardos scheme, for the same parameters ℓ, Z, δ. The distribution F is the same, and the score function S_{ji} is the same. So it is sufficient to prove that in the static Tardos scheme, $\mathbb{P}[S_j > Z] \leq \epsilon_1/2n$. Now, as in the proof of Theorem 8.3, we use the Markov inequality with $\alpha = 1/(d_\alpha c)$ to get

$$\mathbb{P}[S_j > Z] = \mathbb{P}[e^{\alpha S_j} \leq e^{\alpha Z}] \leq e^{-\alpha Z}\mathbb{E}[e^{\alpha S_j}] \leq e^{-\alpha Z + r\alpha^2 \ell}. \tag{9.5}$$

Multiplying both sides of (S2*) by $-c^2 k$ and taking the exponential on both sides gives $e^{-\alpha Z + r\alpha^2 \ell} \leq \epsilon_1/2n$, which proves the result. □

[1] If $[S_j(t_0) > S_j(\ell) > Z]$, then $\{S_j(\ell) \geq Z | S_j(t_0) \geq Z\}$ is satisfied, but $\{S_j(\ell) \geq S_j(t_0)\}$ does not hold. This explains the use of a greater-equal sign instead of an equality sign, giving a probability slightly more than 1/2.

9.4 Special completeness

As was also done in the proof of Soundness, we prove (special) completeness by relating the false negative probability to the false negative probability in the static Tardos scheme. Then we simply use the results from the static Tardos scheme to complete the proof. Below we write $R(t) = \sum_{j \in C} \sum_{i=d(j)+1}^{t} S_{ji}$, where $d(j)$ is the time when user j is disconnected and no longer receives any codewords, and $d(j) = \infty$ if j is never disconnected. We write $S(t) = \sum_{i=1}^{t} \sum_{j \in C: d(j) > i} S_j(t)$, and $\tilde{S}(t) = S(t) + R(t)$ for times t where at least one pirate is still active, and $\tilde{S}(t) = \infty$ otherwise.

Lemma 9.5. *Let C be a coalition of size at most c, and let ρ be any pirate strategy employed by this coalition. Then*

$$\mathbb{P}[C \not\subseteq \sigma(\rho(X))] \leq 2 \cdot \mathbb{P}[\tilde{S}(\ell) < cZ']. \tag{9.6}$$

Proof. If $C \not\subseteq \sigma(\rho(X))$, then at time ℓ at least one pirate is still active, and $S(\ell)$ is trivially at most cZ'. So then $\tilde{S}(\ell) = S(\ell) + R(\ell)$, and $\mathbb{P}[\tilde{S}(\ell) < cZ'] \geq \mathbb{P}_d[S(\ell) < cZ']\mathbb{P}_d[R(\ell) < 0] = \frac{1}{2}$. So if $C \not\subseteq \sigma(\rho(X))$ then with probability at least $1/2$ we have $\tilde{S}(\ell) < cZ'$, giving the result. □

Lemma 9.6. *Let C be a coalition of size at most c, and let ρ be any pirate strategy employed by this coalition. Let* (C1') *and* (C2*) *be satisfied. Then*

$$\mathbb{P}[\tilde{S}(\ell) < cZ'] \leq \frac{\epsilon_2}{2}. \tag{9.7}$$

Proof. First, note that in the dynamic Tardos scheme, the only extra information pirates receive compared to the static Tardos scheme is the fact whether some of them are disonnected from them. This information is certainly covered by the information contained in the values of p_i (if pirates know p_i, then they can calculate their scores S_j themselves and calculate whether they are disconnected or not). Also note that $\tilde{S}(\ell)$ behaves the same as S in the static Tardos scheme, where the total coalition score is calculated for all pirates and all positions, regardless of whether they contributed at that position or not. So if we can prove that even in the static Tardos scheme, and even if coalitions get information about the values of p_i, the probability of keeping the coalition score S below cZ' is bounded by $\epsilon_2/2$, then it follows that also $\mathbb{P}[\tilde{S}(\ell) < cZ'] \leq \epsilon_2/2$.

For the static Tardos scheme, note that the proof method for the completeness property did not use that p_i is secret. The only thing that was used in that proof is that the Marking Assumption applies, which does apply here. So here we can also use the proof method of the static Tardos scheme. So following the first few steps, using the upper bound on the expectation value, and using $Z' - Z < 1/\sqrt{\delta}$ we get

$$\mathbb{P}[S < cZ'] = \mathbb{P}[e^{-\beta S} > e^{-\beta cZ'}] \leq e^{\beta cZ' - g\beta\ell} < e^{\beta cZ + s - g\beta\ell}. \tag{9.8}$$

Multiplying both sides of (C2*) by $-c^2 k\beta$, taking the exponential of both sides, and multiplying both sides by e^s gives $e^{\beta cZ + s - g\beta\ell} \leq e^{-k\eta - \ln(2)} = \epsilon_2/2$, which was to be proven. □

9.5 Optimization

Again, we use the analysis of Blayer and Tassa in [BT08, Section 2.4] and slightly modify it to obtain a new set of optimal parameters for our scheme. The formulas for the optimal values of $d_\delta, d_\alpha, d_z, d_\ell$ in the following Theorem are again almost the same as in Chapter 8.

Theorem 9.7. *Let η, c, k be given, and let r, s, g be fixed, satisfying $r \in (\frac{1}{2}, \infty), s \in (0, \infty), g \in (0, \frac{2}{\pi})$. Let $\eta' = \eta + (s + \ln(2))/k$ and let $\gamma = 1 + \ln(2)/k$. Then the optimal choice of $d_\delta, d_\alpha, d_z, d_\ell$, minimizing d_ℓ and satisfying conditions* (S1),(S2),(C1'),(C2)*, is given by:*

$$\hat{d}_\delta = \left(\frac{1}{\frac{4}{\pi} - 2g} \left(\sqrt{\frac{(h^{-1}(s)s)^2}{c} + \frac{16}{\pi}\left(\frac{2}{\pi} - g\right)} + \frac{h^{-1}(s)s}{\sqrt{c}} \right) \right)^2, \tag{O1}$$

$$\hat{d}_\alpha = \max\left(\frac{\sqrt{\hat{d}_\delta}}{h(r)\sqrt{c}}, \frac{r}{g} + \sqrt{\left(\frac{r}{g}\right)^2 + \frac{r}{g}\frac{\eta'}{\gamma}\sqrt{\frac{\hat{d}_\delta}{s^2 c}}} \right), \tag{O2*}$$

$$\hat{d}_z = \frac{g\hat{d}_\alpha^2\gamma + r\eta'\sqrt{\frac{\hat{d}_\delta}{s^2 c}}}{g\hat{d}_\alpha - r}, \tag{O3*}$$

$$\hat{d}_\ell = \frac{\eta'\sqrt{\frac{\hat{d}_\delta}{s^2 c}} + \hat{d}_z}{g}. \tag{O4*}$$

So to find the optimal septuple $(\hat{r}, \hat{s}, \hat{g}, \hat{d}_\delta, \hat{d}_\alpha, \hat{d}_z, \hat{d}_\ell)$ for given c, η, k, satisfying all requirements and minimizing $\hat{d}_\ell$, one again only has to look for the triple (r, s, g) with $r \in (\frac{1}{2}, \infty), s \in (0, \infty)$ and $g \in (0, \frac{2}{\pi})$ that minimizes the right hand side of (O4*).

Example An optimal solution to (S1),(S2*),(C1'),(C2*) for $c \geq 2$ and $\eta \leq 1$ and $k \geq 5\ln(10) \approx 11.51$, minimizing d_ℓ, is given by

$$d_\ell = 26.16, \quad d_z = 8.64, \quad d_\delta = 26.42, \quad d_\alpha = 4.61, \quad r = 0.66, \quad s = 1.03, \quad g = 0.49. \tag{9.9}$$

This means that with these constants, we can prove soundness and completeness for all $c \geq 2$, $\eta \leq 1$ and $k \geq 11.51$, with a codelength of $\ell = 26.16c^2 \ln(n/\epsilon_1)$.

9.6 Discussion

If we compare our construction with the static Tardos scheme, then we see the biggest advantage is that we now have certainty about catching all pirates, rather than catching at least one pirate. So to catch all pirates, instead of repeating the Tardos scheme c times (giving a total codelength of $\mathcal{O}(c^3 \log(n/\epsilon_1))$), we catch the whole coalition using at most $\mathcal{O}(c^2 \log(n/\epsilon_1))$ time.

Since this scheme is a dynamic scheme, we can also compare it to some of the dynamic traitor tracing schemes we saw earlier. If we compare our scheme to deterministic dynamic schemes, then we see that the time needed is a factor c more. However, we use a much smaller alphabet ($q = 2$). On the other hand, Tassa also used a binary alphabet in his probabilistic dynamic traitor tracing scheme, but compared to that scheme the time/codelength needed for our scheme is a factor $c^2 \log(n)$ less. Furthermore, this construction has some nice properties which no other dynamic traitor tracing scheme has: Some advantages of our construction are as follows.

1. Codewords of users are independent, so it is impossible to frame a specific user.
2. Codeword positions are independent of earlier positions, so for generation of the codewords we do not need the feedback from the pirates.

Especially the second property offers new possibilities. With this scheme, it is possible to generate the whole code and the codewords in advance, and store these codewords at the clientside. The

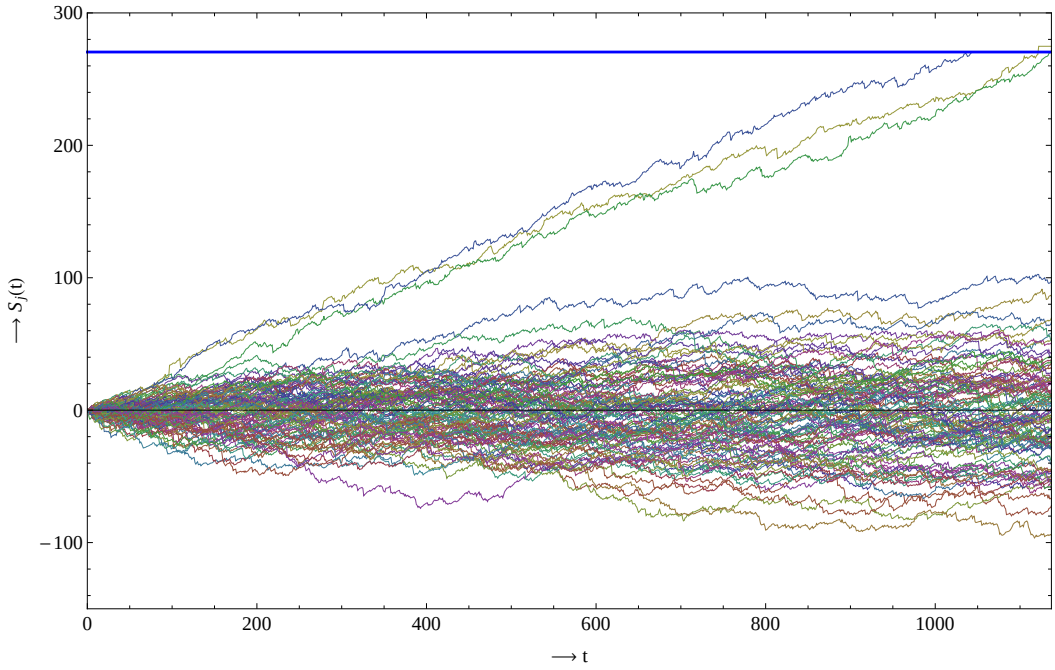

Figure 9.2: *An example of the dynamic Tardos scheme, with* 100 *users,* 3 *of which are pirates. In this case* $Z \approx 270$*, and after* $t \approx 1140$ *time all three pirates have been caught. In this case pirates share the risk, by contributing equally to the pirate broadcast.*

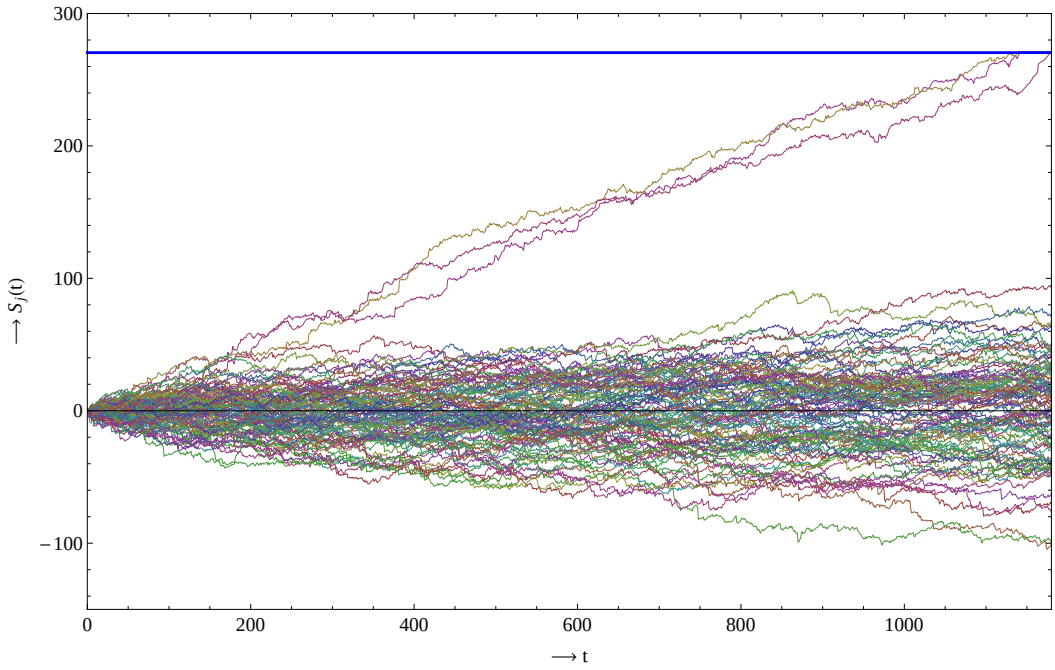

Figure 9.3: *Another example of the dynamic Tardos scheme, with again* 100 *users,* 3 *of which are pirates, and the same scheme parameters. In this case however, the pirates used a different strategy. However, this hardly shows in the result, as again after* $t < 1200$ *time all pirates are caught, and no innocent users are disconnected.*

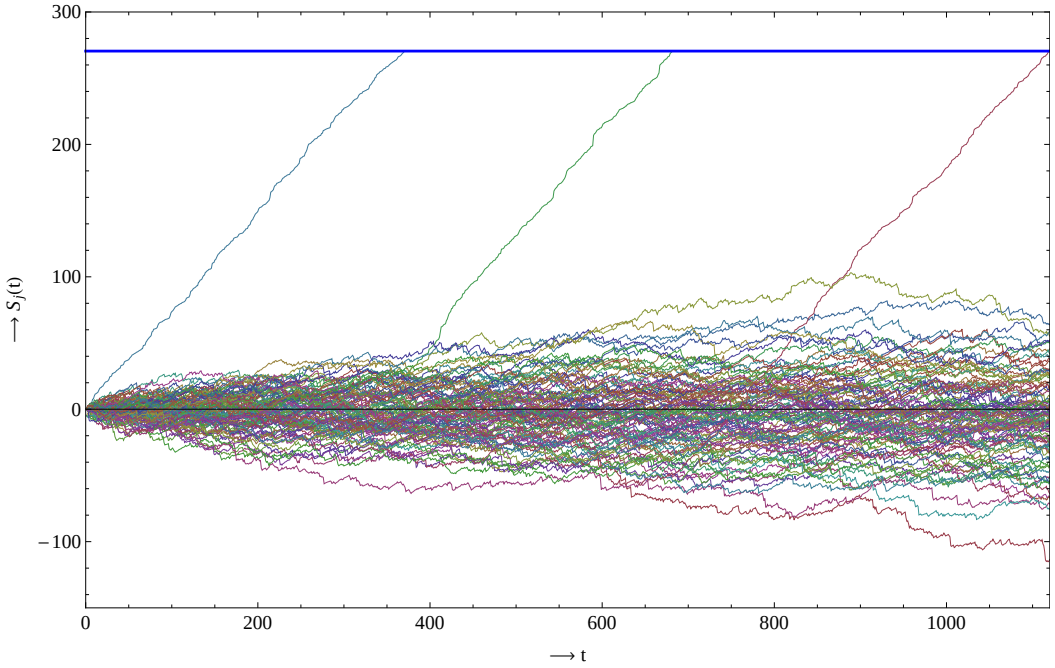

Figure 9.4: *Again we have $n = 100$ users, of which $c = 3$ are pirates. In this case the pirates chose the scapegoat strategy, i.e. sacrificing one pirate, one at a time. This is where the advantage of the dynamic Tardos scheme comes in: In the regular Tardos scheme, the steep line of the first sacrificed pirate would then simply continue all the way until $t = \ell$, giving this user a gigantic score and leaving the other two pirates free of suspicion. However now the first pirate is soon disconnected, after which the other two pirates have to step in. They continue sacrificing pirates one at a time, and soon the second pirate is disconnected. The third pirate finally has to output his data, and he is also disconnected. And this all happens before time $t = 1200$, i.e. compared to the previous two examples the pirates did not gain anything, as again all pirates are disconnected in 1200 symbols.*

scheme only needs to be able to disconnect a user from the system during the process. This is also the only reason why this scheme does not work in a static environment; then the pirates could sacrifice one user for all positions, while here after some steps you want to disconnect that user and force the others to contribute and get their scores up. So the fact that this scheme is in a sense only *semi-dynamic* could give practical advantages over a full dynamic scheme, where codewords depend on the pirate output.

Finally, one does not need to store the codewords or the values of p_i, but only the current accusation scores for each user in this scheme. So the total data storage required for this scheme is constant for each user, even when $\ell, c \to \infty$. But as mentioned above, for practical reasons one may want to store the complete codewords, at the client side or at the side of the distributor.

9.7 Variant

For the above scheme to work, we need to be able to disconnect users after every codeword position. However, we can modify the scheme to also work in a setting where this is not possible, but where we can only disconnect users after each block of b symbols. One solution would be to only use one of the b symbols, so that we need b times as many symbols to disconnect all traitors. But we can do better than that. We can simply use the proof method described above, and see where we need to make changes. As it turns out, we can solve this problem by simply defining $\hat{Z}' = Z + b \max_p S_{ji}(p) < Z + b/\sqrt{\delta}$ as our new upper bound for the scores of users. Then we can follow the whole process described above again, and we finally end up with the following Theorems.

Theorem 9.8. *Let the dynamic Tardos scheme be constructed as above, and let b be the block size such that after every b symbols we can choose to disconnect a user or not. Let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the following two requirements:*

$$d_\alpha \geq \frac{\sqrt{d_\delta}}{h(r)\sqrt{c}}, \tag{S1}$$

$$\frac{d_z}{d_\alpha} - \frac{r d_\ell}{d_\alpha^2} \geq 1 + \frac{\ln(2)}{k}. \tag{S2*}$$

Then the probability of accusing some guilty users is at most ϵ_1, i.e. the scheme is ϵ_1-sound.

Theorem 9.9. *Let the dynamic Tardos scheme be constructed as above, and let b be the block size such that after every b symbols we can choose to disconnect a user or not. Let s, g be positive constants such that d_ℓ, d_z, d_δ, s and g satisfy the following two requirements:*

$$\frac{2 - \frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g, \tag{C1'}$$

$$g d_\ell - d_z \geq \left(\eta + \frac{bs + \ln(2)}{k}\right)\sqrt{\frac{d_\delta}{s^2 c}}. \tag{C2\#}$$

Then the probability of not accusing all guilty users is at most ϵ_2, i.e. the scheme is ϵ_2-complete.

To find an optimal solution, one can again apply use formulas similar to those in Theorem 9.7, but with $\eta' = \eta + (s + \ln(2))/k$ replaced by $\hat{\eta}' = \eta + (bs + \ln(2))/k$.

Example Let $b = 10$. An optimal solution to (S1),(S2*),(C1'),(C2#) for $c \geq 2$ and $\eta \leq 1$ and $k \geq 5\ln(10) \approx 11.51$, minimizing d_ℓ, is given by

$$d_\ell = 33.57, \quad d_z = 9.46, \quad d_\delta = 18.70, \quad d_\alpha = 4.92, \quad r = 0.62, \quad s = 0.82, \quad g = 0.48. \tag{9.10}$$

This means that with these constants, we can prove soundness and completeness for all $c \geq 2$, $\eta \leq 1$, $k \geq 11.51$, and $b \leq 10$ with a codelength of $\ell = 33.57c^2 \ln(n/\epsilon_1)$.

Note that for very large b, we basically end up in a static scenario where we can only disconnect users after all of the symbols have been distributed.

9.8 Summary

In this chapter we gave a new construction of a probabilistic dynamic traitor tracing scheme based on the Tardos scheme, which uses a codelength/time of at most $\mathcal{O}(c^2 \ln(n/\epsilon_1))$. Furthermore, as described in the Discussion, this scheme has several other advantages, making it a very suitable choice for use in practice. The proofs are heavily based on the proofs of the Tardos scheme, so most of the work was already done. Finally we also discussed a variant of the scheme, which may be useful in case disconnecting users after every symbol is impossible or ¨expensive¨. Even if we can only disconnect users after every block of b symbols, we still get very good and short codelengths.

Chapter 10

The universal Tardos scheme

10.1 Introduction

In the second part of the Tardos Quadrilogy, we saw that we can construct an efficient probabilistic dynamic traitor tracing scheme based on the Tardos scheme. However, for this scheme we needed an input c in advance, as the Tardos scheme depends on c. So one natural question is: Can we construct a dynamic Tardos scheme that works even when c is not known in advance? The answer is yes, as we will see in this chapter. The idea is to first move the dependence on c from the traitor tracing code to the score function, and then run several instances of the dynamic Tardos scheme (for different values of c) simultaneously, all using the same symbols. We will show that then we can catch any coalition of any a priori unknown size c in at most $\mathcal{O}(c^2 \log(n/\epsilon_1))$ time. This is a huge improvement over Tassa's scheme [Tas05] described in Chapter 7.

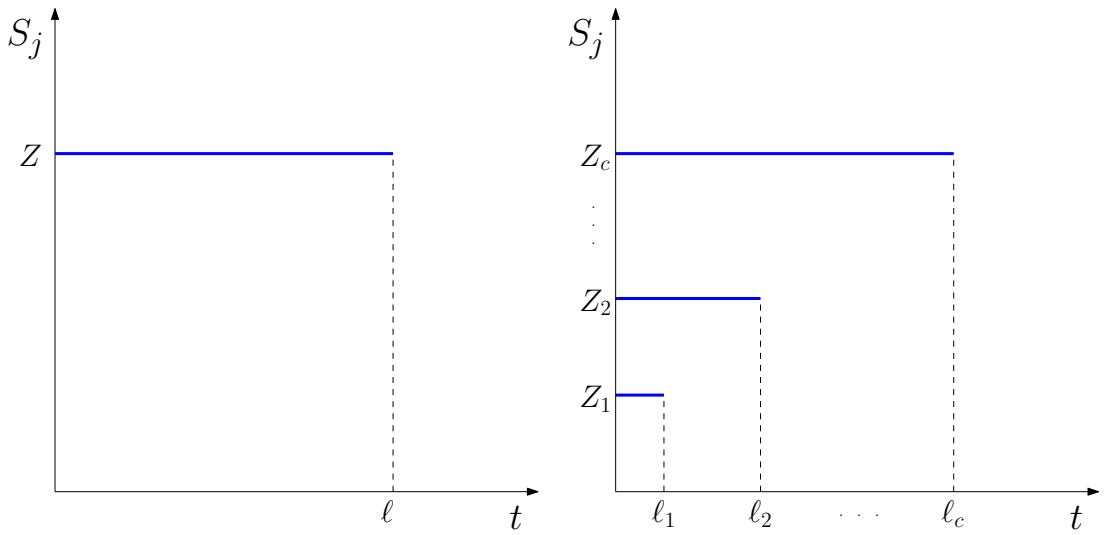

Figure 10.1: *The main idea of the universal Tardos scheme is to run multiple instances of the dynamic Tardos scheme simultaneously, so that one gets multiple blue lines which will get a user disconnected. On the left we see the schematic representation of the dynamic Tardos scheme for a fixed c, while on the right we see the schematic construction of the universal Tardos scheme, for collusions of size at most c. Each block of width ℓ_i and height Z_i corresponds to one instance of the dynamic Tardos scheme, with $c = i$.*

Let us briefly go back to the original symmetric Tardos scheme. Looking closely, we see that there is a chain of dependencies which make the codewords dependent on c: We take $X_{ji} \sim \mathrm{Ber}(p_i)$, we take $p_i \sim F$, the function F depends on δ', δ' depends on δ, and we take $\delta = 1/(300c)$. So

the values of X_{ji} actually depend on the value of c. Furthermore for the proof of soundness we use that δ is large enough, so that $\alpha S_{ji} \leq \alpha\sqrt{1/\delta}$ can be bounded from above, while e.g. for the proof of completeness we use several times that δ is sufficiently small, so we cannot simply take δ fixed regardless of c.

For the solution to our problems, we turn our attention to the probability density function used for the values p_i. For fixed c, let us denote this function by $f_c(p)$. Let us recall the definition of $f_c(p)$:

$$f_c(p) = \frac{1}{(\pi - 4\delta'_c)\sqrt{p(1-p)}}. \tag{10.1}$$

The important observation now is that this function $f_c(p)$ is actually almost the same for all values of c; for different c we merely use a different scaling of this function to make sure the integral sums up to 1. In particular, writing $f_\infty(p) = \lim_{c\to\infty} f_c(p) = \frac{1}{\pi\sqrt{p(1-p)}}$, which is a function that does not depend on c, we get the relation:

$$f_c(p) = \frac{\pi}{\pi - 4\delta'_c} f_\infty(p). \tag{10.2}$$

The function $f_c(p)$ and the term $\pi - 4\delta'_c$ are chosen such that integrating f_c from δ_c to $1-\delta_c$ gives exactly 1, i.e. such that this is a probability density function. Using the above relation, this means that integrating f_∞ from δ_c to $1-\delta_c$ gives

$$\int_{\delta_c}^{1-\delta_c} f_\infty(p) = \frac{\pi - 4\delta'_c}{\pi} = 1 - \frac{4\delta'_c}{\pi}. \tag{10.3}$$

Since $\delta'_c = \arcsin(\sqrt{\delta_c}) = \arcsin(\sqrt{1/(300c)})$ is very small, integrating $f_\infty(p)$ from δ_c to $1-\delta_c$ gives roughly 1. So the area below the curve $f_\infty(p)$ between the offsets δ_c and $1-\delta_c$ is close to 1.

Now the solution to a c-independent traitor tracing scheme is as follows. We simply use the function f_∞ for our codeword generation. Then, during the accusation phase, we calculate scores for each user for each value of c. For any c, we need that the values of p_i were taken according to f_c. Therefore for any value of c we simply disregard all positions for which p_i was not in the range $[\delta_c, 1-\delta_c]$. Above we calculated the cost of this, and saw that we are actually throwing away only a small percentage of the data. For the positions that are counted, the values of p_i are actually according to the distribution f_c, since f_c is just a rescaled version of f_∞ between the offsets δ_c and $1-\delta_c$. The mechanics for the dynamic Tardos scheme thus remain the same, and the proofs will go analogously.

10.2 Construction

Let us now give the construction in full, with a full explanation of what is going on afterwards. Let $n \geq 2$ be a positive integer, and let $\epsilon_1, \epsilon_2 \in (0,1)$ be the desired upper bounds for the soundness and completeness error probabilities respectively. Let $\vec{v}(i)$ be the characteristic vector such that $v_c = 1$ if $p_i \in [\delta_c, 1-\delta_c]$ and $v_c = 0$ otherwise. Then the universal Tardos traitor tracing scheme works as follows.

1. **Initialization**
   (a) Take some vector $\vec{\kappa}$ such that $\sum_{c=1}^{\infty} \kappa_c \leq 1$.
   (b) Take the vector $\vec{k}$ according to $k_c = \ln(n/\kappa_c\epsilon_1)$.

(c) Take parameters $\vec{\lambda}, \vec{\zeta}, \vec{d}$ such that for some parameters $\vec{\alpha}, \vec{\rho}, \vec{\gamma}, \vec{\sigma}$, for each c the following requirements are satisfied:

$$\alpha_c \geq \frac{\sqrt{d_c}}{h(\rho_c)\sqrt{c}}, \tag{US1}$$

$$\frac{\zeta_c}{\alpha_c} - \frac{\rho_c \lambda_c}{\alpha_c^2} \geq 1 + \frac{\ln(2)}{k_c}, \tag{US2}$$

$$\frac{2 - \frac{4}{d_c}}{\pi} - \frac{h^{-1}(\sigma_c)\sigma_c}{\sqrt{d_c c}} \geq \gamma_c, \tag{UC1}$$

$$\gamma_c \lambda_c - \zeta_c \geq \left(\eta + \frac{\sigma_c + \ln(2)}{k_c}\right)\sqrt{\frac{d_c}{\sigma_c^2 c}}. \tag{UC2}$$

(d) Take the codelength vector $\vec{\ell}$ according to $\ell_c = \lambda_c c^2 k_c$.

(e) Take the accusation offset vector $\vec{Z}$ according to $Z_c = \zeta_c c k_c$.

(f) Take the cutoff vector $\vec{\delta}$ according to $\delta_c = 1/(d_c c)$.

(g) Set the initial score vector $\vec{S}_j$ for each user j according to $(S_j)_c = 0$ for all c.

(h) Set the initial time vector $\vec{t}$ according to $t_c = 0$.

2. **Generation/Distribution/Accusation**
For each time $i \geq 1$ do the following.

(a) Select $p_i \in [0, 1]$ from the distribution defined by the functions $F_\infty(p)$ and $f_\infty(p)$:

$$F_\infty(p) = \frac{2\arcsin(\sqrt{p})}{\pi}, \quad f_\infty(p) = \frac{1}{\pi\sqrt{p(1-p)}}. \tag{10.4}$$

(b) For each user j, select X_{ji} according to $\mathbb{P}[X_{ji} = 1] = 1 - \mathbb{P}[X_{ji} = 0] = p_i$.

(c) Send to each connected user j the ith symbol X_{ji}.

(d) Intercept the pirate output y_i, or terminate if there is none.

(e) Calculate the scores for position i, according to:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ -\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 0, \\ +\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 0. \end{cases} \tag{10.5}$$

(f) Update the scores of each user according to $\vec{S}_j := \vec{S}_j + S_{ji}\vec{v}_i$.

(g) Update the times according to $\vec{t} := \vec{t} + \vec{v}_i$.

(h) If $(S_j)_c > Z_c$ and $t_c \leq \ell_c$ for some index c, then disconnect user j.

Let us explain a bit more what we are actually doing here. First, we take some vector $\vec{\kappa}$ which must sum to at most one. The purpose of these constants is the following. For each c, we will bound the false positive probability by what is inside the logarithm of k_c, i.e. $\kappa_c \cdot \epsilon_1/n$. However, since we run all these schemes simultaneously, the total false positive probability for a single user is bounded from above by summing over all false positive probabilities for each c. So the probability that an innocent user is ever accused is bounded by $\sum_{c=1}^{\infty} \kappa_c \epsilon_1/n = \epsilon_1/n \sum_{c=1}^{\infty} \kappa_c$. Since we want this to be bounded from above by ϵ_1/n, we get the requirement that $\sum_{c=1}^{\infty} \kappa_c \leq 1$. One way to realize this is to take e.g. $\kappa_c = (1/2)^c$, so that κ_c decreases exponentially in c. However,

then we get $k_c = \ln(n/\kappa_c\epsilon_1) = \ln(2^c n/\epsilon_1) = \mathcal{O}(c\ln(n/\epsilon_1))$, so that $\ell_c = \mathcal{O}(c^3\ln(n/\epsilon_1))$ which is not what we want. Fortunately we can also take e.g. $\kappa_c = 6/\pi^2c^2$ (using that $\sum_{i=1}^{\infty} 1/c^2 = \pi^2/6$), so that $k_c = \mathcal{O}(\ln(n/\epsilon_1))$ and $\ell_c = \mathcal{O}(c^2\ln(n/\epsilon_1))$. If c is expected to be small, then to make $\ell_1, \ell_2, \ell_3, \ldots$ as small as possible, it is better to take $\kappa_c = \mathcal{O}(1/c^N)$ for some large N, so that most of the weight is at the beginning. However, if c may be large, then $\kappa_c = \mathcal{O}(1/c^{1+\epsilon})$ may be a better choice, so that for large c, κ_c is not that small. So there is an obvious tradeoff here, and the optimal solution depends on the scenario.

Now let us continue with the construction. The constants k_c play the role of $k = \ln(n/\epsilon_1)$ in the previous chapters, except now there is this extra term inside the logarithm, making the values of k different for each c. The parameters $\vec{\lambda}, \vec{\zeta}, \vec{d}, \vec{\alpha}, \vec{\rho}, \vec{\gamma}, \vec{\sigma}$ play the roles of $d_\ell, d_z, d_\delta, d_\alpha, r, g, s$ respectively, except that now these are also vectors. The renaming is done for convenience, to avoid double indices. Note that (US1), (US2), (UC1), (UC2) are the same as (S1), (S2*), (C1'), (C2*) from the previous chapter, only with variables renamed. Next we also take ℓ_c, Z_c, δ_c for each c differently, using these variables $\lambda_c, \ldots, \sigma_c$. Finally we initialize all scores for all users at 0, and the counter of used positions for each c is set at 0. These counters t_c will count how many of the p_is up to now were between δ_c and $1-\delta_c$, i.e. how many positions were not discarded for this value of c.

Then comes the actual distribution/accusation phase. For each time, we first generate a value p_i according to F_∞ (which does not depend on c). This p_i is then used to generate symbols X_{ji}, as is usual in the Tardos scheme. The symbols are distributed, and if a pirate transmitter is still active, we assume we will intercept some pirate output y_i[1]. Then, for each user j we calculate the value S_{ji} (which in fact also does not depend on c), but for updating the scores we now only increase those scores $(S_j)_c$ for which $p_i \in [\delta_c, 1-\delta_c]$. This is done simply by adding $S_{ji} \cdot \vec{v}_i$, since $\vec{v}_i$ has the nice property of indicating for which values of c the scores should be updated. Similarly, the counters are updated simply by adding $\vec{v}_i$ to $\vec{t}$ (adding 1 only for those c for which these symbols were used), and for each user j and coalition size c we check whether the cth score of user j exceeded the cth threshold Z_c. Note that in most cases, $(S_j)_c \approx (S_j)_{c+1}$ while Z_c and Z_{c+1} may be quite far apart.

After this, the process repeats for the next value of i, generating new symbols, distributing them, updating scores and disconnecting users. The process terminates if, as mentioned above, no pirate output is received anymore, but again, depending on the application, one may want to have different rules for termination (e.g. after a fixed number of positions the process simply has to stop).

10.3 Results

Let us now formally prove results above this scheme. Using the above construction, we get the following results, which can be easily proved using results from previous chapters.

Theorem 10.1. *Let the universal Tardos scheme be constructed as above. Then the probability of ever accusing an innocent user is at most ϵ_1/n, hence the probability of never disconnecting any innocent users is at least $1-\epsilon_1$.*

Proof. We chose the parameters $\lambda_c, \ldots, \sigma_c$ such that they satisfy the requirements from the dynamic Tardos scheme with parameter c and error probability $\epsilon_{1,c} = \kappa_c\epsilon_1$. Hence for each c we

[1] If no output is received, then we are happy. Either we can wait and repeat the same symbols until output is received (which only means the pirates have lost part of the content for their distribution), or we can at some point terminate, concluding that we must have caught all pirates. Of course this also depends on the scenario, e.g. if before the pirate output stopped no user was disconnected, then a pirate is still active and one may want to continue to wait.

know that the probability of having $(S_j)_c > Z_c$ before the time i when $t_c = \ell_c$ is at most $\kappa_c \epsilon_1 / n$. So the probability that the user is ever accused is bounded from above by:

$$\mathbb{P}[j \in \sigma] \leq \sum_{c=1}^{\infty} \kappa_c \frac{\epsilon_1}{n} \leq \frac{\epsilon_1}{n}. \tag{10.6}$$

The proof can again be completed by noting that $(1 - \epsilon_1/n)^n \geq 1 - \epsilon_1$. □

Theorem 10.2. *Let the universal Tardos scheme be constructed as above. Let C be a coalition of some a priori unknown size c. Then the probability that by the time i when $t_c = \ell_c$ some members of the coalition are still active is bounded from above by ϵ_2.*

Proof. We chose the parameters $\lambda_c, \ldots, \sigma_c$ such that they satisfy the requirements from the dynamic Tardos scheme with parameter c, so the result follows from the proofs given in the previous Chapter. □

Theorem 10.3. *Let the universal Tardos scheme be constructed as above. Let T_c be the time at which we see the ℓ_cth value of p_i between $[\delta_c, 1 - \delta_c]$. Then $T_c - \ell_c$ is distributed according to a negative binomial distribution, with parameters $r = \ell_c$ and $p = \frac{4}{\pi}\delta'_c$. Hence T_c has mean $\mu = \ell_c/(1-p)$ and variance $\sigma^2 = \ell_c p/(1-p)^2$, and $\mathbb{P}[T_c \geq \mu + a]$ for $a > 0$ decreases exponentially in a.*

Proof. The fact that $T_c - \ell_c$ follows a negative binomial distribution can be easily verified by checking the definition of the negative binomial distribution, while we are waiting for $r = \ell_c$ successes, with each success happening with probability $p = 1 - \mathbb{P}(p_i \in [\delta_c, 1 - \delta_c]) = \frac{4}{\pi}\delta'_c$. Finally the mean and variance of the negative binomial, as well as the size of the tails, are well known. □

To summarize the results, we see that the number of symbols T_c needed to reach ℓ_c useful symbols is a random variable with mean $\ell_c/(1 - \frac{4}{\pi}\delta'_c)$ and exponentially small tails, and one we reach this time T_c, we know that with probability $1 - \epsilon_2$ we will have caught all guilty users of the coalition, if the coalition had size at most c. Furthermore the probability of ever disconnecting an innocent user is at most ϵ_1.

Example Let us use the sequence $\kappa_c = 6/(\pi^2 c^2)$, such that $\sum_{c=1}^{\infty} \kappa_c = 1$. Then for fixed c we get $\ell_c = \lambda_c c^2 \ln(nc^2\pi^2/6\epsilon_1)$, so for constant λ_c we get $\ell_c = \mathcal{O}(c^2 \ln(n/\epsilon_1))$. Let $n = 1000$ and $\epsilon_1 = \epsilon_2 = 0.01$. Then we can take the parameters as in Tables 10.1 and 10.2, giving codelengths $\ell_c = \mathcal{O}(c^2 \ln(n/\epsilon_1))$.

Example Again let us use the sequence $\kappa_c = 6/(\pi^2 c^2)$. Let $n = 100$ and $c = 3$, and let $\epsilon_1 = 0.01$ and $\epsilon_2 = 0.5$. Figures 10.2, 10.3, 10.4, 10.5 graphically show a simulation of this situation, where the pirates used a pirate-symmetric strategy. The graphs show the scores of users set against the time, with different lines corresponding to different users. However, since we have to keep multiple scores for each user, every user now also has multiple lines. But in most cases these lines overlap, while Figure 10.2 shows one of the rare exceptions where the different scores for each user are actually different.

c	k_c	ℓ_c	Z_c	δ_c
2	13.3969	1385	229.433	0.0187624
3	14.2079	2772	329.503	0.0122054
4	14.7832	4567	427.245	0.0089459
5	15.2295	6753	523.522	0.0070081
6	15.5941	9319	618.780	0.0057293
7	15.9024	12258	713.285	0.0048252
8	16.1695	15562	807.211	0.0041540
9	16.4051	19227	900.678	0.0036371
10	16.6158	23249	993.774	0.0032275
15	17.4267	48610	1455.52	0.0020273
20	18.0021	82535	1913.66	0.0014505
25	18.4484	124860	2369.97	0.0011159
30	18.8130	175479	2825.32	0.0008991
35	19.1213	234317	3280.18	0.0007482
40	19.3884	301320	3734.83	0.0006376
45	19.6240	376445	4189.45	0.0005534
50	19.8347	459660	4644.17	0.0004873

Table 10.1: *The parameters for the universal Tardos scheme, for* $n = 1000$ *and* $\epsilon_1 = \epsilon_2 = 0.01$*, using the sequence* $(\kappa_c) = (6/\pi^2 c^2)$*. Since* $k_c = \ln(n/\kappa_c\epsilon_1)$ *with* κ_c *decreasing in* c*,* k_c *is increasing in* c*. To catch a coalition of size* 50*, we roughly need at most* 460000 *usable symbols, which is less than* 58 *kB. Note that it is also clear from the table that* Z_c *grows linearly in* c *(*$Z_c = \mathcal{O}(ck_c)$*), while* ℓ_c *grows quadratically in* c *(*$\ell_c = \mathcal{O}(c^2 k_c)$*).*

c	λ_c	ζ_c	d_c	α_c	ρ_c	σ_c	γ_c
2	25.8332	8.56291	26.6491	4.60329	0.663136	1.03032	0.486256
3	21.6741	7.73053	27.3104	4.13024	0.647677	1.10635	0.498434
4	19.3054	7.22518	27.9456	3.84325	0.637387	1.16136	0.506936
5	17.7352	6.87511	28.5383	3.64457	0.629786	1.20459	0.513419
6	16.5994	6.61338	29.0904	3.49612	0.623819	1.24024	0.518626
7	15.7300	6.40768	29.6066	3.37950	0.618941	1.27062	0.522958
8	15.0374	6.24022	30.0917	3.28459	0.614838	1.29708	0.526654
9	14.4691	6.10027	30.5498	3.20531	0.611313	1.32055	0.529868
10	13.9920	5.98090	30.9841	3.13770	0.608233	1.34162	0.532703
15	12.3972	5.56816	32.8850	2.90412	0.597011	1.42349	0.543242
20	11.4618	5.31510	34.4700	2.76105	0.589638	1.48224	0.550334
25	10.8289	5.13861	35.8468	2.66134	0.584245	1.52816	0.555600
30	10.3639	5.00597	37.0735	2.58644	0.580043	1.56590	0.559745
35	10.0034	4.90131	38.1862	2.52736	0.576629	1.59796	0.563136
40	9.71326	4.81581	39.2086	2.47912	0.573773	1.62584	0.565989
45	9.47304	4.74415	40.1575	2.43870	0.571330	1.65050	0.568440
50	9.26981	4.68288	41.0450	2.40414	0.569204	1.67263	0.570580

Table 10.2: *The parameters* $\vec{\lambda} \ldots \vec{\sigma}$ *for the universal Tardos scheme with* $n = 1000$ *and* $\epsilon_1 = \epsilon_2 = 0.01$*, satisfying the four conditions and minimizing* λ_c *for every* c*. Note again that as in the previous chapter,* λ_c *converges to* $\pi^2/2 \approx 4.93$*,* ζ_c *converges to* $\pi \approx 3.14$*,* α_c *converges to* $\pi/2 \approx 1.57$*,* ρ_c *converges to* 0.5 *and* γ_c *converges to* $2/\pi \approx 0.64$*, while* d_c *and* σ_c *will (slowly) diverge.*

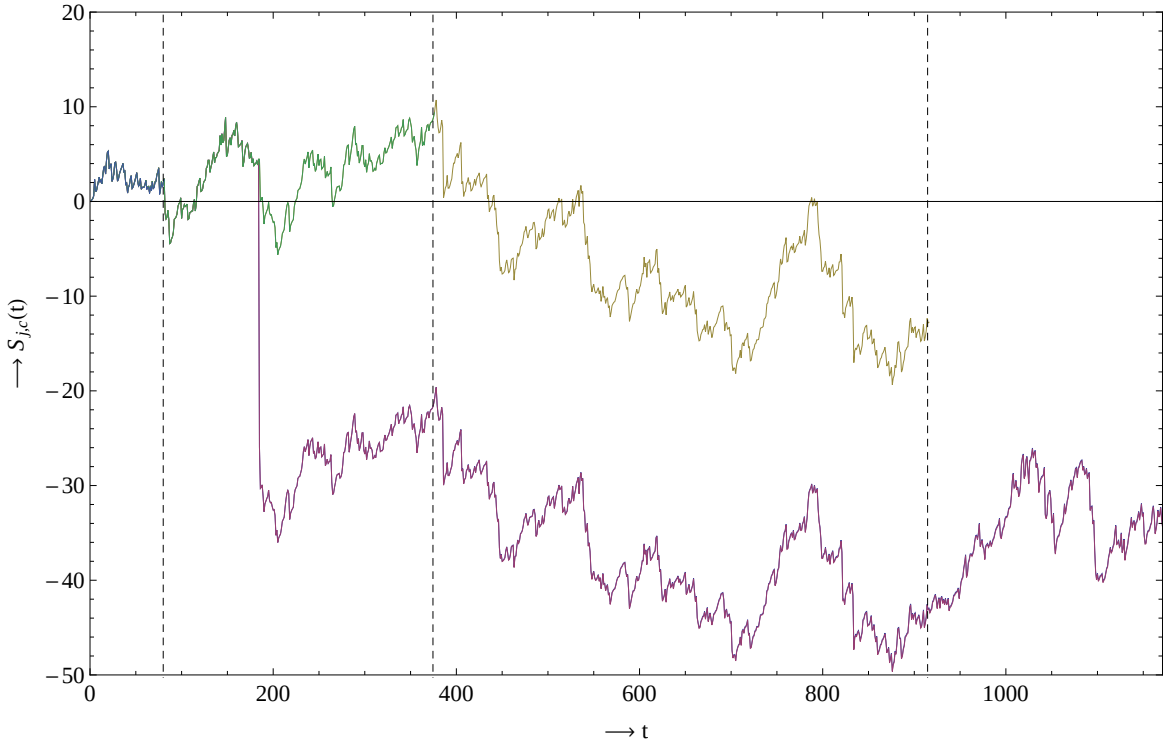

Figure 10.2: *An example of the multiple scores for a single user. The dashed lines correspond to the times when $t_1 = \ell_1$, $t_2 = \ell_2$ and $t_3 = \ell_3$, while the thresholds Z_1, Z_2, Z_3, Z_4 are not in range and therefore are not shown here. In this case, around time $t \approx 185$, we had a rare event; the value p_i was either in the interval $[\delta_4, \delta_3]$ or in the interval $[1-\delta_3, 1-\delta_4]$, and although an extreme percentage of users received the same symbol here, the user received the other, unlikely symbol. Since the pirates outputted the most common symbol, the score of the user dropped drastically for $c \geq 4$, while for $c = 2, 3$ this position was disregarded (due to the fact that $p_i \notin [\delta_3, 1-\delta_3]$). Note that this is a very rare event; we had to run several simulations on a hundred users to get one of these rare events to occur. In most cases, the graphs for a fixed user, for different values of c, are almost indistinguishable, since on positions which are disregarded for some c, all users usually receive the same symbol.*

10.4 Discussion

So the scheme we saw above is able to catch coalitions of any a priori unknown size c in $\mathcal{O}(c^2 \ln(n/\epsilon_1))$ time, with arbitrary high probability. This is already a huge improvement over earlier results from Tassa [Tas05]. However, this is not all, as this scheme has many more advantages.

First of all, as we also saw with the dynamic Tardos scheme, the code is independent of the pirate output. The only thing we use the pirate output for is to disconnect users inbetween. This means that we could theoretically generate the whole vector p_i and the whole code matrix X in advance, instead of inbetween rounds. This means we will never have to worry about the time between receiving pirate output and sending new symbols, as this can be done instantly. Also, this means that one could try to somehow store the part of the matrix belonging to user j (i.e. $\vec{x}_j$) at the client side of user j, instead of distributing symbols one at a time. If this can somehow be made secure, so that users cannot tamper with their codewords, then this would save the distributor of having to send symbols to each user over and over. Instead, he could then send the whole codeword at the start, and then start the process of distributing content and disconnecting users. This could be a real advantage of this scheme, as private messages to each user are generally costly.

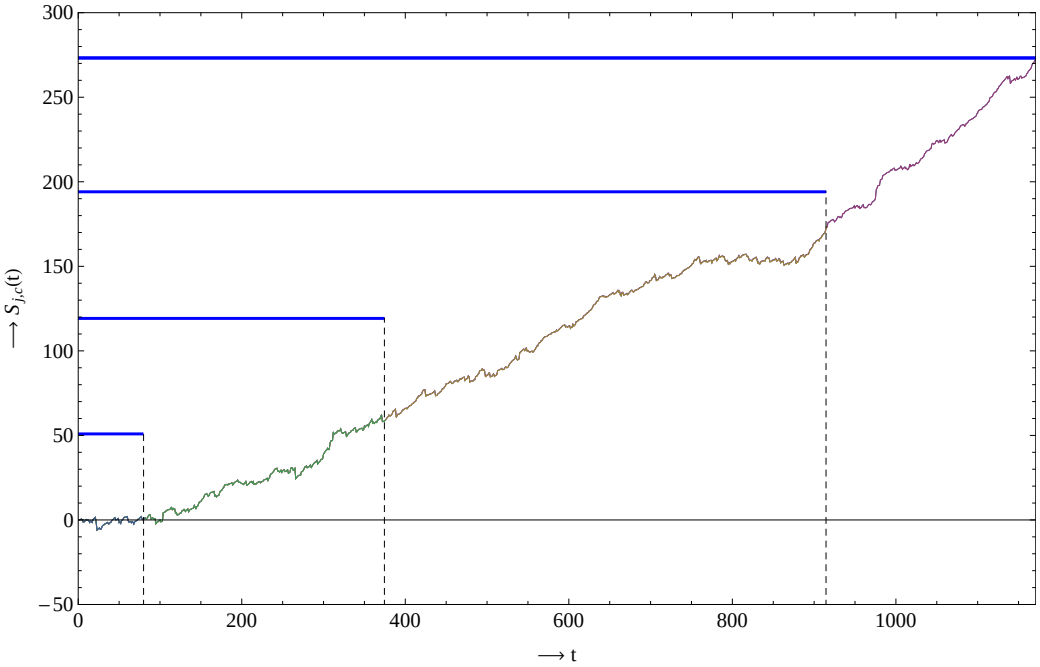

Figure 10.3: *The scores of a single pirate over time. The blue straight lines correspond to the thresholds Z_c, while the colored lines correspond to the scores for different values of c. The score for some $c = i$ only exists on the interval $[0, \ell_i)$, e.g. the green line for $c = 2$ starts at $t = 0$ (but overlaps with the blue line on the interval $[0, \ell_1)$) and stops at the second dashed line, around $t \approx 370$. At $t \approx 1160$ this pirate hits the ceiling and is disconnected from the system.*

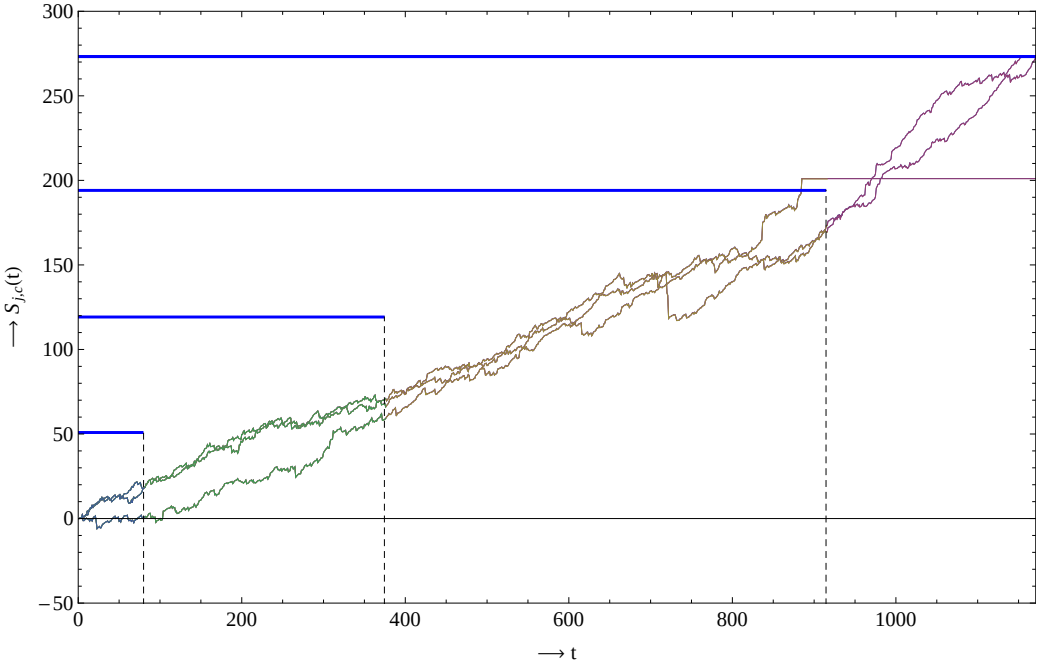

Figure 10.4: *The scores of all three pirates in the system. Note that in this case, the scheme failed to find and disconnect all pirates before time ℓ_3, as two pirates are still active then (although their scores are close to the threshold Z_3). This means that this simulation of the scheme falls in the ϵ_2-tail of exceptions, as the probability of not catching all pirates by time ℓ_3 is at most ϵ_2 (in this case $\epsilon_2 \approx 1/2$). Also note that as the pirate that is caught first does not receive any symbols anymore, his score remains constant after that.*

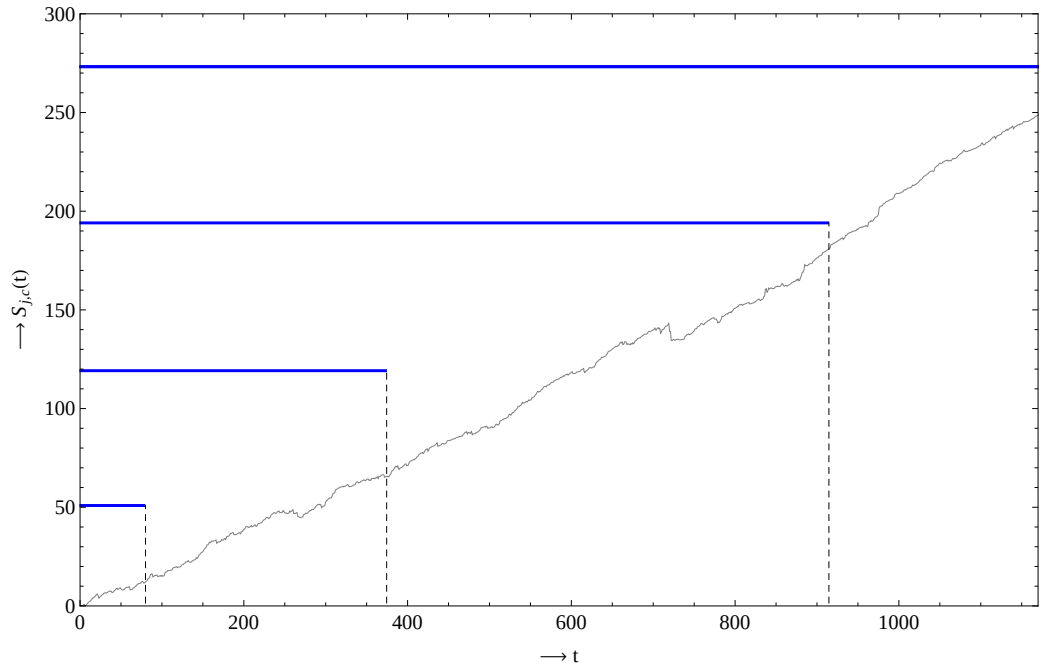

Figure 10.5: *The average pirate score (i.e. $A(t) = \frac{1}{|C|}\sum_{j\in C} S_j(t)$) over time. Regardless of the strategy used, this line will be close to linear in t, which intuitively shows that the scheme will eventually catch all pirates, as the thresholds only grow as $\mathcal{O}(\sqrt{t})$. So even if the scheme fails to catch all c pirates before time ℓ_c, with even higher probability all c pirates will be caught before time ℓ_{c+1}.*

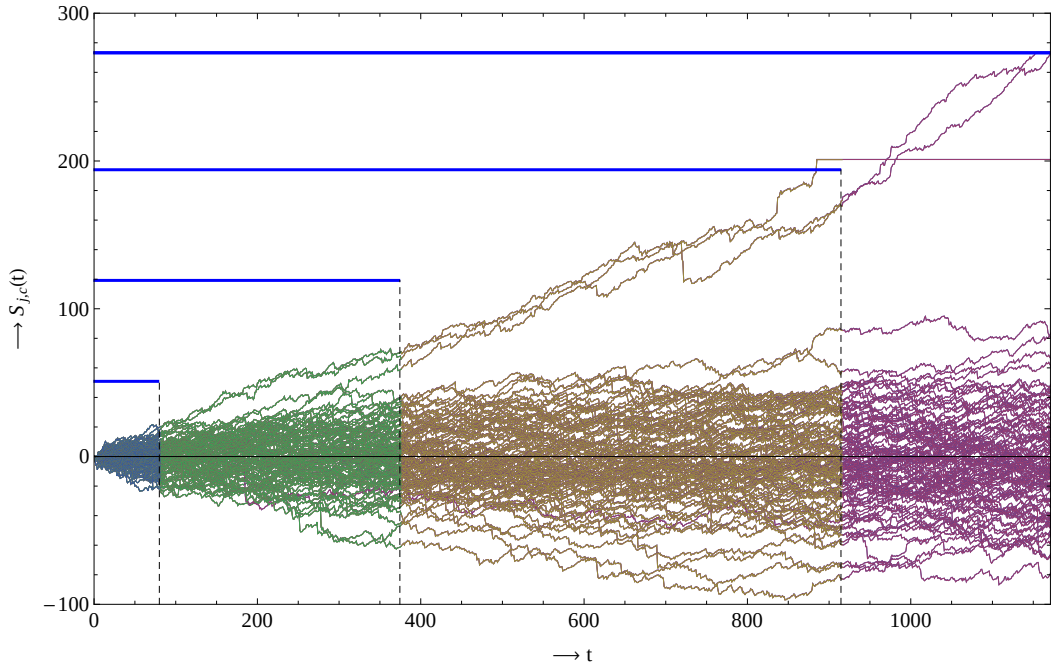

Figure 10.6: *All scores of all* 100 *users in the system. If you look very closely, you will see a bit of purple around* $(200, -40)$ *and* $(700, -50)$*, which corresponds to the one user whose different scores did not completely overlap (see Figure 10.3). Notice that innocent users' scores are quite far below the thresholds Z_c, and behave like simple one-dimensional random walks. The cloud of scores will get wider and wider over time (also with width roughly $\mathcal{O}(\sqrt{t})$), but Z (as a function of c) is such that it grows fast enough over time to avoid ever hitting innocent users with high probability.*

Secondly, note that the whole construction is basically identical for every time i. The symbols are always generated using the same distribution function f_∞, and the score function never changes either. So in fact, if e.g. at some time i_0 a second pirate broadcast is detected, one could start a second universal Tardos scheme, running simultaneously with the first one. Both traitor tracing algorithms could use the same symbols for their score functions, and both coalitions can be traced simultaneously. The probability that an innocent user is accused in one of the two schemes is then bounded by $2\epsilon_1$ rather than ϵ_1, but this can be solved by simply taking $\epsilon_1' = \epsilon_1/2$. One could start generalizing this, and make statements like any set of coalitions (with cardinality constant in c, n) of size at most c can be traced in $\mathcal{O}(c^2 \ln(n/\epsilon_1))$ time, taking ϵ_1' as ϵ_1 divided by the cardinality of the set of coalitions. In any case, this shows that we can trace multiple coalitions simultaneously, even if the pirate broadcasts do not start at the same time.

Thirdly, note that for some fixed values of c and ϵ_1, we get some threshold value Z_c and a length ℓ_c to use for this dynamic Tardos scheme. If however we used a different value ϵ_1', we would have had a different value of Z_c and a different codelength ℓ_c, but the process would be the same. This means that in our scheme, before time $\ell_c(\epsilon_1')$ we could also check whether user scores exceed $Z_c(\epsilon_1')$. In other words: besides running the dynamic Tardos scheme for each c, for some fixed ϵ_1, we could also simultaneously run the dynamic Tardos scheme for each c, for some other fixed ϵ_1'. Here we do get in trouble when really running these schemes simultaneously (since you have to decide whether you disconnect a suspect or not), but one could use these other thresholds $Z_c(\epsilon_1')$ to calculate some sort of probability that a user is guilty. First the pirate would cross a 90% barrier (i.e. the probability that innocent users cross this line is $< 10\%$), then a 95% barrier, and when he crosses a 99% barrier he is disconnected. Then already before the user is disconnected, we can give a statistic to indicate the 'suspiciousness' of this user. If a user then does not cross the final barrier, one could still decide whether to disconnect him later.

Finally, another advantage of this scheme is another consequence of the fact that the scheme is identical for every i, namely that we can concatenate several instances of this process to form one larger process. For example, suppose one movie is broadcast, and during the tracing process for this movie no users or only few users are caught. Then the pirates remain active, and when another movie is broadcast (possibly soon after, or only weeks after) they could start broadcasting again. By initializing the scores of users by the scores they had at the end of the first movie (and also loading the counters t_c), one could start the tracing process with the pirates probably already having a pretty high score. So then the pirates will sooner hit the roof and be disconnected, than if we had to start over with scores 0 for everyone.

10.5 Summary

The idea for a universal dynamic Tardos scheme consists of two ideas. First, the scheme is based on running several dynamic Tardos schemes simultaneously, all using the same code. So instead of needing $\sum_c \ell_c = \sum_c \mathcal{O}(c^2) = \mathcal{O}(c^3)$ symbols to run all dynamic Tardos schemes up to a fixed value c sequentially, we re-use symbols from smaller values of c for larger coalitions and run all schemes simultaneously. Then secondly, we noted that the distribution functions f_c are actually quite similar for different values of c; the difference is only a scaling factor, and by disregarding 'bad values of p' from the distribution f_∞, the remaining symbols are exactly distributed according to f_c. So we can in fact run such schemes simultaneously, if we only disregard part of the data for fixed values of c.

Since all the hard work was already done in earlier chapters to prove that the dynamic Tardos scheme is secure, the proofs in this chapter then followed easily. The result is that we have a fully dynamic scheme based in the Tardos scheme, which can trace a coalition of any unknown size c using at most $\mathcal{O}(c^2 \ln(n/\epsilon_1))$ symbols. This improves upon results by Tassa (Chapter

7) by a factor c^2, i.e. a huge improvement. After that we saw that this scheme also has some advantages w.r.t. flexibility, which can be used by the distributor in his implementation of the traitor tracing scheme.

Summarizing, we can say that the scheme discussed in this chapter is very interesting, both theoretical and practical.

Chapter 11

The staircase Tardos scheme

11.1 Introduction

In the third part of the Tardos Quadrilogy, we saw that we can convert the dynamic Tardos scheme for fixed c into a universal scheme for finding any coalition of any size. That is, if a coalition has size c_0, then with high probability the scheme will find and disconnect all c_0 traitors in $\mathcal{O}(c_0^2 \ln(n/\epsilon_1))$ time, rather than $\mathcal{O}(c^2 \ln(n/\epsilon_1))$ time, where c is some large upper bound to the number of colluders. Using a universal distribution function which can be used for all Tardos schemes, and then running several dynamic Tardos schemes simultaneously, we saw we are able to achieve this goal.

In this chapter, we investigate a different transition from the dynamic Tardos scheme to a universal Tardos scheme, where only one scheme has to be run simultaneously. So with this scheme, the advantage is that we only have to keep one score per user. Also, since here we do not disregard any data, we can prove that after a fixed number of positions ℓ_c, with probability $1 - \epsilon_2$ we will have caught all colluders, if a coalition has size c or less (unlike in the universal Tardos scheme, where due to an increase in the codelength (which itself is a random variable) the codelength is not really fixed). The price we pay however may be higher than the gains described above, as now the scheme does not use a single distribution function f_∞ for the whole interval $[0, \infty)$ anymore, but a function f_1 for the interval $[0, \ell_1)$ and functions f_{i+1}^* for the intervals $[\ell_i, \ell_{i+1})$. This means that the scheme is less flexible; for instance, we cannot concatenate different schemes anymore, since the distribution functions f_1 and f_i^* are essentially different.

So how does the scheme work, and what are these functions f_c^*? Well, to make the Tardos proof mechanism work, we do not need the same distribution of the values of p on the whole interval $[0, \ell)$, as the proof even works if the values of p are known to the users. In particular, sorting the values of p, permuting the values of p in any way, or using a distribution f_A on $[0, m)$ and a distribution f_B on $[m, \ell_c)$ such that on average we get f_c, will not cause the proof method to fail. Here we use the latter, by using different distribution functions on different intervals. All we need is that in total, the distribution is $f_c(p)$.

The construction is now as follows. On the interval $[0, \ell_1)$ (i.e. for catching coalitions of size 1) we use the distribution function $f_1(p)$, so that on $[0, \ell_1)$, the values of p_i are distributed according to $f_1(p)$. For the interval $[\ell_1, \ell_2)$, we use some distribution function $f_2^*(p)$ for the values of p_i, such that on the complete interval $[0, \ell_2)$ the values are distributed according to $f_2(p)$. Similarly, on $[\ell_2, \ell_3)$ we use $f_3^*(p)$, and more generally on $[\ell_c, \ell_{c+1})$ we use $f_{c+1}^*(p)$, such that on $[0, \ell_{c+1})$ the values of p are distributed according to $f_{c+1}(p)$.

So how do we choose these functions $f_{c+1}^*(p)$? For this we simply use that we want that taking the

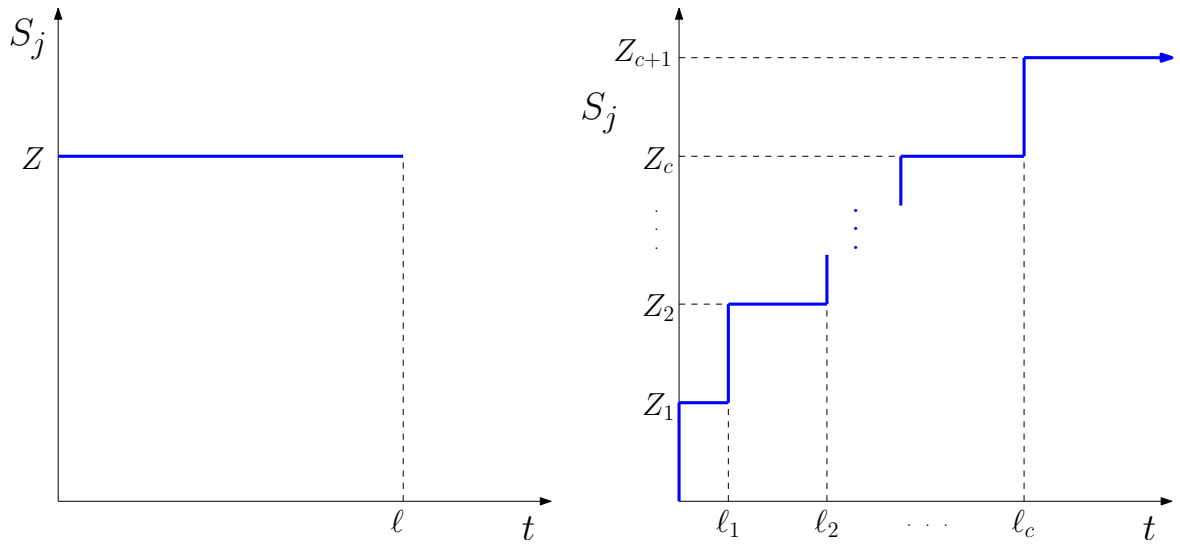

Figure 11.1: *A sketch of the transition from the dynamic Tardos scheme (left) to the staircase Tardos scheme (right). The blue line again indicates the threshold, above which users are accused and disconnected from the system. The scheme lends its name from the 'staircase' on the right, which roughly grows as $\mathcal{O}(\sqrt{t})$ over time (since Z grows linear in c, and ℓ grows quadratic in c). Every user has one score function, which has to stay below the staircase for them to remain connected.*

union of $[0, \ell_c)$ with distribution f_c and $[\ell_c, \ell_{c+1})$ with distribution f^*_{c+1} gives an interval $[0, \ell_{c+1})$ with distribution f_{c+1}. In other words, we want that $\ell_c \cdot f_c(p) + (\ell_{c+1} - \ell_c) \cdot f^*_{c+1}(p) = \ell_{c+1} \cdot f_{c+1}(p)$ for each p. Isolating f^*_{c+1} in this equation, we get

$$f^*_{c+1}(p) = \frac{\ell_{c+1} f_{c+1}(p) - \ell_c f_c(p)}{\ell_{c+1} - \ell_c}. \tag{11.1}$$

An important question is: Is this a well-defined distribution function? One can check that indeed, this is all well-defined for any p, and the only non-trivial requirement we should check is that $f^*_{c+1}(p) \geq 0$ for all p. As it is a safe assumption to say that $\ell_{c+1} - \ell_c > 0$, this comes down to the question whether $\ell_{c+1} f_{c+1} > \ell_c f_c$ for any c and p. In other words: Is $h_p(c) = \ell_c f_c(p)$ an increasing function of c, for any p?

First, recall that $\ell_c = K_1 c^2$ for some constant $K_1 > 0$ not (or only very slightly) depending on c, and $f_c(p) = p^{-1/2}(1-p)^{-1/2}(\pi - K_2 c^{-4/3})^{-1}$ for some constant $K_2 > 0$ not (or only very slightly) depending on c. Filling this in in $h_p(c) = \ell_c f_c(p)$ gives

$$h_p(c) = \frac{K_1}{\sqrt{p(1-p)}} \cdot \frac{c^2}{\pi - K_2 c^{-4/3}}, \tag{11.2}$$

$$h'_p(c) = \frac{K_1}{\sqrt{p(1-p)}} \cdot \frac{2c\pi - \frac{10}{3} K_2 c^{-1/3}}{(\pi - K_2 c^{-4/3})^2}. \tag{11.3}$$

For large c it is obvious that $h'_p(c)$ is positive, and the numerator of the second fraction of $h'_p(c)$ is increasing in c, hence $h'_p(c) > 0$ for all $c \geq 2$ if $h'_p(2) > 0$, which one can easily verify. So indeed, $h_p(c)$ is increasing in c for any p, and $f^*_{c+1}(p)$ is a well-defined distribution function on $[\delta_{c+1}, 1 - \delta_{c+1}]$.

Finally, we have not yet discussed the threshold Z. For this, we now use the staircase, such that on the interval $[0, \ell_1)$ its height is Z_1, and on $[\ell_c, \ell_{c+1})$ its height is Z_{c+1}. This will again guarantee that innocent users will stay below the line, while Z grows slowly enough for any coalition score to cross this threshold.

11.2 Construction

Let us now give the complete construction of the staircase Tardos scheme. Let $n \geq 2$ be a positive integer as usual, and let $\epsilon_1, \epsilon_2 \in (0,1)$ be the desired upper bounds for the soundness and completeness error probabilities respectively. Then the staircase Tardos fingerprinting scheme works as follows.

1. **Initialization**
   (a) Take the vector $\vec{\kappa}$ such that $\sum_{c=1}^{\infty} \kappa_c \leq 1$.
   (b) Take the vector $\vec{k}$ according to $k_c = \ln(n/\kappa_c \epsilon_1)$.
   (c) Take parameters $\vec{\lambda}, \vec{\zeta}, \vec{d}$ such that for some parameters $\vec{\alpha}, \vec{\rho}, \vec{\gamma}, \vec{\sigma}$, for each c the following requirements are satisfied:

$$\alpha_c \geq \frac{\sqrt{d_c}}{h(\rho_c)\sqrt{c}}, \tag{S1}$$

$$\frac{\zeta_c}{\alpha_c} - \frac{\rho_c \lambda_c}{\alpha_c^2} \geq 1 + \frac{\ln(2)}{k_c}, \tag{S2}$$

$$\frac{2 - \frac{4}{d_c}}{\pi} - \frac{h^{-1}(\sigma_c)\sigma_c}{\sqrt{d_c c}} \geq \gamma_c, \tag{C1}$$

$$\gamma_c \lambda_c - \zeta_c \geq \left(\eta + \frac{\sigma_c + \ln(2)}{k_c}\right) \sqrt{\frac{d_c}{\sigma_c^2 c}}. \tag{C2}$$

   (d) Take the codelength vector $\vec{\ell}$ according to $\ell_c = \lambda_c c^2 k_c$.
   (e) Take the accusation offset Z according to $Z = \zeta_1 k_1$.
   (f) Take the cutoff vector $\vec{\delta}$ according to $\delta_c = 1/(d_c c)$.
   (g) Take the initial score S_j for each user j according to $S_j = 0$ for all c.

2. **Generation/Distribution/Accusation**
   For each time $i \geq 1$, let c' be such that $i \in [\ell_{c'-1}, \ell_{c'})$ (with $\ell_0 = 0$), let $Z = Z_{c'} = \zeta_c k_c$, and do the following.
   (a) Select $p_i \in [0,1]$ from the distribution function $f^*_{c'}(p)$:

$$f^*_{c'}(p) = \frac{1}{\ell_{c'} - \ell_{c'-1}} \left(\frac{\ell_{c'}}{\pi - 4\delta'_{c'}} - \frac{\ell_{c'-1}}{\pi - 4\delta'_{c'-1}} \right) \frac{1}{\sqrt{p(1-p)}}. \tag{11.4}$$

   (b) For each user j, select X_{ji} according to $\mathbb{P}[X_{ji} = 1] = 1 - \mathbb{P}[X_{ji} = 0] = p_i$.
   (c) Send to each connected user j the ith symbol X_{ji}.
   (d) Intercept the pirate output y_i, or terminate if there is none.
   (e) Calculate the scores for position i, according to:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ -\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 0, \\ +\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 0. \end{cases} \tag{11.5}$$

   (f) Update the score of each user according to $S_j := S_j + S_{ji}$.
   (g) If $S_j > Z$ then disconnect user j.

11.3 Results

For the staircase Tardos scheme described above, we get the following results.

Theorem 11.1. *Let the staircase Tardos scheme be constructed as above. Then the probability of ever accusing an innocent user is at most ϵ_1/n, hence the probability of never disconnecting any innocent users is at least $1 - \epsilon_1$.*

Proof. We chose the parameters $\lambda_c, \ldots, \sigma_c$ such that they satisfy the requirements from the dynamic Tardos scheme with parameter c and error probability $\epsilon_{1,c} = \kappa_c \epsilon_1$. Hence for each c we know that the probability of having $S_j > Z_c$ before time ℓ_c is at most $\kappa_c \epsilon_1 / n$. So the probability that the user is ever accused is bounded from above by ϵ_1/n, and the proof is completed by noting that $(1 - \epsilon_1/n)^n \geq 1 - \epsilon_1$. □

Theorem 11.2. *Let the staircase Tardos scheme be constructed as above. Let C be a coalition of some a priori unknown size c. Then the probability that by time ℓ_c some members of the coalition are still active is bounded from above by ϵ_2.*

Proof. We chose the parameters $\lambda_c, \ldots, \sigma_c$ such that they satisfy the requirements from the dynamic Tardos scheme with parameter c, so the result follows from the proofs given in that chapter. □

Example We again use the sequence $\kappa_c = 6/(\pi^2 c^2)$. Let $n = 100$ and $c = 10$, and let $\epsilon_1 = \epsilon_2 = 0.01$. Figures 11.2, 11.3, 11.4 graphically show a simulation of this situation, where the pirates used a pirate-symmetric strategy. The graphs show the scores of users set against the time, with different lines corresponding to different users. With high probability, we expect to catch all colluders before reaching the 10th step of the staircase, and in this case we even catch the last pirates at the 6th step of the stairs.

11.4 Summary

Again this chapter was relatively short, as most of the work had already been done before. The idea of this chapter is to extend the dynamic Tardos scheme to a universal Tardos scheme in a different way, such that we only have to keep one score for each user. To do this however, we needed to update the distribution functions for the values of p at every time ℓ_c. How to update this distribution was quite trivial, as it is clear which result we want to get. After that the results again followed easily.

In general, the universal Tardos scheme will be more practical, as the memory usage for scores of users is not an issue, while the universal Tardos scheme is more flexible than this scheme in several ways. However, this staircase Tardos scheme has its (theoretic) charms as well, using only one score for each user, and using all data for each c. And of course, if the added flexibility is not needed or not useful, then this staircase Tardod scheme is still slightly more efficient, as for any fixed c the time at which ℓ_c is reached here (simply $t = \ell_c$) is smaller than the time at which ℓ_c is reached in the universal Tardos scheme ($t > \ell_c$).

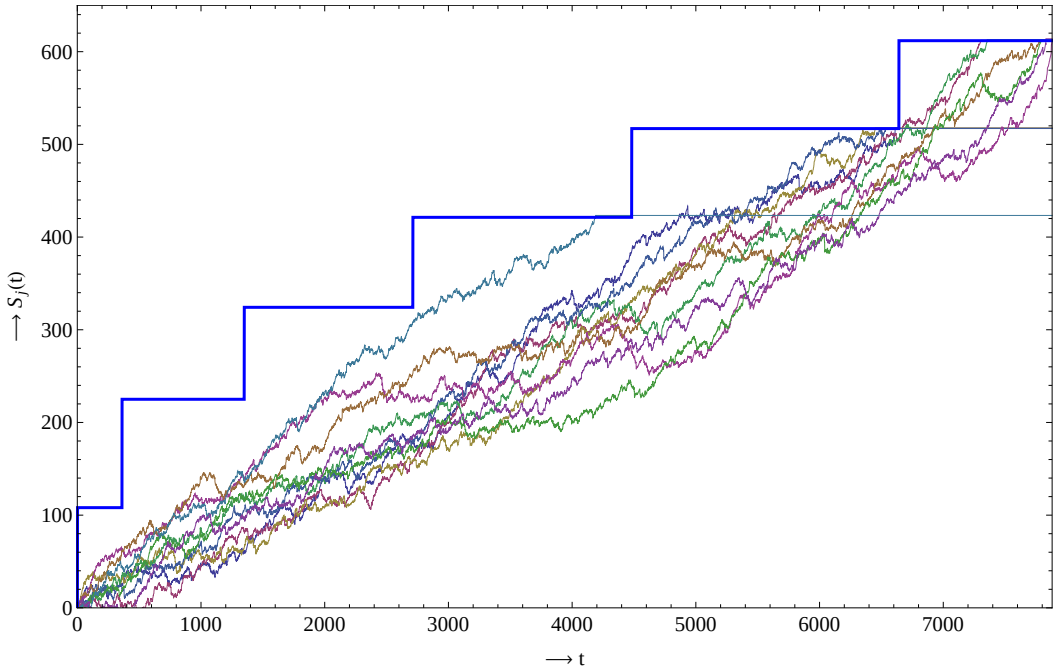

Figure 11.2: *The scores of all pirates over time. The blue staircase corresponds to the increasing value of Z, and the different lines correspond to different users. In this case there are* 10 *pirates, and even before $t = \ell_6$ we have already caught all pirates. One pirate was already disconnected before time ℓ_4, and one pirate was disconnected before time ℓ_5.*

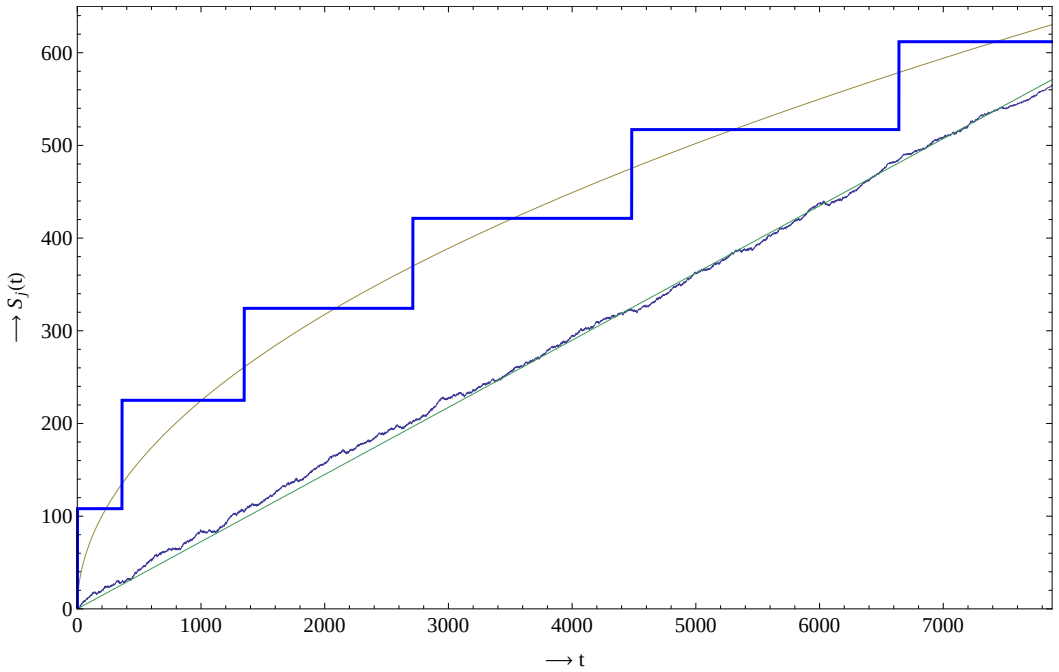

Figure 11.3: *The average pirate score (i.e. $A(t) = \frac{1}{|C|}\sum_{j\in C} S_j(t)$) over time. Regardless of the strategy used, this line is close to linear in t, which intuitively shows that the scheme will eventually catch all pirates, as the thresholds only grow as $\mathcal{O}(\sqrt{t})$. The straight line and the curve are put in to further emphasize this asymptotic behaviour of the scores and thresholds. Even if the scheme fails to catch all pirates before time ℓ_c, then with even higher probability all pirates are caught before time ℓ_{c+1}.*

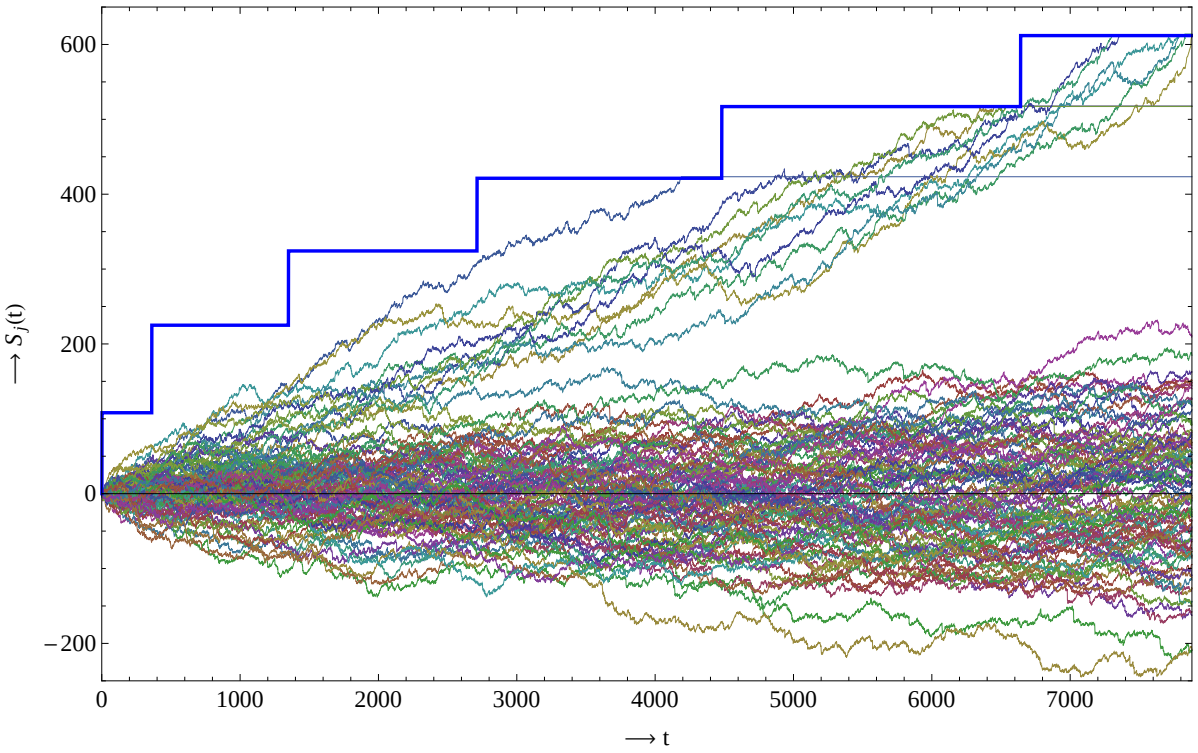

Figure 11.4: *All scores of all* 100 *users in the system. Note that innocent users' scores are quite far below the threshold* $Z(t)$*, and behave like simple one-dimensional random walks. The cloud of scores will get wider and wider over time (with width roughly* $\mathcal{O}(\sqrt{t})$*, but* Z *grows fast enough over time to avoid ever hitting innocent users with high probability.*

Chapter 12

Publications

As we believe that the results from Part II are of scientific value, the result of this project is not only this report. Most of the other products are not yet completed, but we mention them here anyway.

12.1 Paper: Optimal symmetric Tardos traitor tracing codes

Since the results in Chapter 8 improve upon results from Skoric et al. [SKC08], Nuida et al. [NFH$^+$09] and Blayer and Tassa [BT08], we decided to write a paper on the results from Chapter 8. Since Skoric et al., Nuida et al. and Blayer and Tassa all published their work in the journal Designs, Codes and Cryptography, we plan to submit it there as well. This will happen not long after this project is finished, as the paper is nearly completed. But of course we cannot yet guarantee that the paper will be accepted there, or when it will be published. A preliminary version of this paper can be found in Appendix A.

12.2 Paper: Dynamic Tardos traitor tracing

Another plan for the future is to write a paper about Chapters 9, 10 and 11, i.e. about dynamic traitor tracing schemes based on the Tardos scheme. Since the results of these chapters are both of scientific and practical value, and improve upon the results from e.g. Tassa [Tas05] by a factor c^2, it should not be a problem to get this paper accepted somewhere, once it is finished. However this paper is far from finished, and we also do not give a preliminary version here yet.

12.3 Irdeto Patent

Since the results about dynamic Tardos traitor tracing are of practical use to Irdeto, they decided to file a patent regarding these results. More details about this patent may follow later.

Chapter 13

Conclusion

13.1 Comparison

To summarize and compare the schemes discussed in this report, we have put together a table containing the most important schemes, their alphabet sizes and their codelengths. Of course many other aspects play a role as well when choosing which scheme to use, but these two parameters are often the two most important factors. The result can be found in Table 13.1. Note that in this table, we have used big-Oh notation for the codelengths, and have disregarded constants. Also, we did not mention any of the Tardos scheme variants and improvements here, besides our own contributions from Part II, as all Tardos variants have the same order codelength of $\mathcal{O}(c^2 \ln(n/\epsilon))$.

To compare schemes more closely, we can also focus on binary (and therefore probabilistic) schemes only, and not disregard constants. Table 13.2 shows how several schemes mentioned in this report compare in that sense. Here we did mention the constants, for small c and for asymptotically large c.

From the last table it is obvious that especially the dynamic schemes presented in this report improve upon literature by a large factor. The scheme of Tassa has a codelength (time) which is roughly the square of the codelength (time) used for our schemes. As for the contribution from Chapter 8, the asymptotic codelength is better than that of any other known Tardos variant, but for small c Nuida et al. have better results.

13.2 Summary

In Part I we gave an extensive overview of many schemes found in literature. We discussed many results we found in a big pile of articles, which we then sorted in four categories of traitor tracing schemes. Hopefully Part I will save some people, who want to get acquainted with results about collusion-resistant traitor tracing schemes, some work, by proving a structured overview of many of these results. However, I do not claim this list is incomplete. In fact, since I only had 9 months to do all this work, I had to make some choices of which articles I should and should not discuss in this report. In particular, the scheme of Jin and Lotspiech [JLN04], which is used in Blu-ray, and the broadcast scheme of Fiat and Naor [FN94,NNL01] are not discussed here, partly because they did not seem to fit with the categorization used in Part I. Also, some miscellaneous schemes for fixed small coalition sizes (e.g. one suggested by Sebe [SDF02], which was found to be broken by Schaathun [Sch08] if users are not forced to contribute equally to the fingerprint, and one suggested by Schaathun [Sch03]) are not discussed here, due to time limitations. But

All schemes	q (alphabet)	ℓ, t (codelength,time)
Deterministic static schemes	$q \geq c+1$	$\ell \geq \Omega(c^2 \log(n))$
- Staddon et al.	$q = \mathcal{O}(c^2 \log(n))$	$\ell = \mathcal{O}(c^2 \log(n))$
- Alon et al.	$q = c+1$	$\ell = \mathcal{O}(c^2 \log(n))$
Probabilistic static schemes	$q \geq 2$	$\ell \geq \Omega(c^2 \ln(n/\epsilon_1))$
- Boneh and Shaw (cubic)	$q = 2$	$\ell = \mathcal{O}(n^3 \ln(n/\epsilon_1))$
- Boneh and Shaw (quartic)	$q = 2$	$\ell = \mathcal{O}(c^4 \ln(n/\epsilon_1) \ln(c/\epsilon_1))$
- Tardos	$q = 2$	$\ell = \mathcal{O}(c^2 \ln(n/\epsilon_1))$
- Laarhoven, De Weger (Ch. 8)	$q = 2$	$\ell = \mathcal{O}(c^2 \ln(n/\epsilon_1))$
Deterministic dynamic schemes	$q \geq c+1$	$t \geq \Omega\left(\frac{c^2}{q-c} + c \log_c(n)\right)$
- Fiat and Tassa	$q = 2c+1$	$t = \mathcal{O}(c \log(n))$
- Berkman et al. (clique)	$q = c+1$	$t = \mathcal{O}(c^3 \log(n))$
- Berkman et al. (degree)	$q = c+1$	$t = \mathcal{O}(c^3 \log(n))$
- Berkman et al. (optimal)	$q = c+1$	$t = \mathcal{O}(c^2 + c \log(n))$
Probabilistic dynamic schemes	$q \geq 2$	?
- Tassa	$q = 2$	$\ell \cdot t = \mathcal{O}(c^4 \log(n) \ln(c/\epsilon_1))$
- Laarhoven et al. (Ch. 9)	$q = 2$	$t = \mathcal{O}(c^2 \ln(n/\epsilon_1))$
- Laarhoven et al. (Ch. 10)	$q = 2$	$t = \mathcal{O}(c^2 \ln(nc^2/\epsilon_1))$
- Laarhoven et al. (Ch. 11)	$q = 2$	$t = \mathcal{O}(c^2 \ln(nc^2/\epsilon_1))$

Table 13.1: *A comparison of schemes presented in this report. The four headers correspond to the four Chapters 4, 5, 6 and 7 respectively, with the exception of the new contributions from Part II.*

still, we hope that the overview from Part I is somewhat complete, and understandably explains the different schemes.

Then, in Part II we gave four new suggestions for improved traitor tracing schemes, all based on the Tardos scheme. We argued that our contributions are in fact good and noteworthy, and improve upon the best results from literature either by a small margin (Chapter 8, compared to Nuida et al.), or by a large margin (Chapter 10, compared to Tassa). As we saw in the previous chapter, we think the results are even good enough to publish them in scientific magazines, and Irdeto even patented the results based on dynamic Tardos traitor tracing. However, both papers and the patent are still a work in progress, and some of it will only finish after this report is already finished. Only the paper based on the improvements from Chapter 8 is almost done, and a preliminary version is given in the Appendix.

13.3 Future work

Let us finally mention some directions for future work, based on things that either the author of this report did not have the time for or simply could not solve. Some of these problems may be extensive enough to keep a new Master student busy for 9 months, while some (other) problems may be solved in a few hours by smart people.

Probabilistic binary schemes	ℓ, t (for small c)	ℓ, t (for large c)
Static schemes	$\ell \geq \Omega(c^2 \ln(n/\epsilon_1))$	$\ell \geq 1.38c^2 \ln(n/\epsilon_1)$
- Boneh and Shaw (cubic)	$\ell \approx 2n^3 \ln(n/\epsilon_1)$	$\ell \approx 2n^3 \ln(n/\epsilon_1)$
- Boneh and Shaw (quartic)	$\ell \approx 32c^4 \ln(n/\epsilon_1) \ln(c/\epsilon_1)$	$\ell \approx 32c^4 \ln(n/\epsilon_1) \ln(c/\epsilon_1)$
- Tardos	$\ell = 100c^2 \ln(n/\epsilon_1)$	$\ell = 100c^2 \ln(n/\epsilon_1)$
- Vladimirova et al.	$\ell \approx 90c^2 \ln(n/\epsilon_1)$	$\ell \approx 39.48c^2 \ln(n/\epsilon_1)$
- Blayer and Tassa	$\ell \approx 85c^2 \ln(n/\epsilon_1)$	$\ell \approx 19.74c^2 \ln(n/\epsilon_1)$
- Skoric et al.	$\ell \approx 50c^2 \ln(n/\epsilon_1)$	$\ell \approx 9.87c^2 \ln(n/\epsilon_1)$
- Nuida et al.	$\ell \approx 5c^2 \ln(n/\epsilon_1)$	$\ell \approx 5.35c^2 \ln(n/\epsilon_1)$
- Laarhoven, De Weger (Ch. 8)	$\ell \approx \mathbf{24}c^2 \ln(n/\epsilon_1)$	$\ell \approx \mathbf{4.93}c^2 \ln(n/\epsilon_1)$
- Amiri and Tardos (impractical)	$\ell = 1c^2 \ln(n/\epsilon_1)$	$\ell \approx 1.39c^2 \ln(n/\epsilon_1)$
Dynamic schemes	?	?
- Tassa	$\ell \cdot t = \mathcal{O}(c^4 \log(n) \ln(c/\epsilon_1))$	$\ell \cdot t = \mathcal{O}(c^4 \log(n) \ln(c/\epsilon_1))$
- Laarhoven et al. (Ch. 9)	$t \approx \mathbf{26}c^2 \ln(n/\epsilon_1)$	$t \approx \mathbf{4.93}c^2 \ln(n/\epsilon_1)$
- Laarhoven et al. (Ch. 10)	$t \approx \mathbf{26}c^2 \ln(nc^2/\epsilon_1)$	$t \approx \mathbf{4.93}c^2 \ln(nc^2/\epsilon_1)$
- Laarhoven et al. (Ch. 11)	$t \approx \mathbf{26}c^2 \ln(nc^2/\epsilon_1)$	$t \approx \mathbf{4.93}c^2 \ln(nc^2/\epsilon_1)$

Table 13.2: *A comparison of binary probabilistic traitor tracing schemes. Besides the schemes of Boneh and Shaw, and Tassa, all schemes in the table are based on the Tardos scheme. Note that the terms c^2 in the logarithms of the last two rows are in fact arbitrary; the power of c depends on the κ_c used. If c is expected to be small, then a larger power of c may be more efficient, while for large c one would get a power $c^{1+\lambda}$ with λ small.*

13.3.1 The Tardos scheme: Discrete distribution functions

Nuida et al. [NFH+09] investigated a version of the Tardos scheme, where the continuous distribution function F (for selecting values p_i) was replaced a discrete distribution function, based on Legendre polynomials. With such distributions they prove that for small c the codelength can really be reduced further than the lower bound of $\pi^2/2$ as in the Tardos scheme with continuous F. So certainly for small c there are advantages of using this approach. Unfortunately I did not have the time to thoroughly investigate this paper in my research.

However, one can easily establish that indeed, for small c and different (discrete) f, it is possible to obtain much shorter codelengths than in the regular Tardos construction. Suppose e.g. $c = 2$ and we use the Tardos scheme with $f(1/2) = 1$ (i.e. $p_i = \frac{1}{2}$ for all i). Let us write $\mathrm{RW}(n)$ for a standard random walk $X = \sum_{i=1}^{n} X_i$, where $\mathbb{P}[X_i = \pm 1] = \frac{1}{2}$. Obviously innocent users' scores $S_j(t)$ behave like standard random walks, while for a coalition of size 2 we have $S_i = +2$ if both colluders have the same symbol and $S_i = 0$ if they receive different symbols. So $S(t) - t$ behaves like a standard random walk as well. Using the Chernoff bound for these standard random walks we get:

$$\mathbb{P}[j \in \sigma(\rho(X))] = \mathbb{P}[\mathrm{RW}(\ell) > Z] \leq e^{-Z^2/2\ell} \leq \epsilon_1/n, \tag{13.1}$$

$$\mathbb{P}[\sigma(\rho(X)) \cap C = \emptyset] \leq \mathbb{P}[\mathrm{RW}(\ell) < 2Z - \ell] \leq e^{-(\ell-2Z)^2/2\ell} \leq \epsilon_2. \tag{13.2}$$

Note that (13.1) and (13.2) only hold if $Z > 0$ (i.e. $d_z > 0$) and $2Z < \ell$ (i.e. $d_z < d_\ell$). Filling in $k = \ln(n/\epsilon_1), \ell = d_\ell c^2 k, Z = d_z c k, \eta = \frac{\ln(1/\epsilon_2)}{\ln(n/\epsilon_1)}$ and $c = 2$ we get $d_z^2 \geq 2d_\ell$ and $4(d_\ell - d_z)^2 \geq 2\eta d_\ell$.

The optimal solution is there where both inequalities are equalities:

$$d_z = 2 + \sqrt{\eta}, \tag{13.3}$$

$$d_\ell = \frac{(2+\sqrt{\eta})^2}{2} \tag{13.4}$$

Hence for $\eta = 1$ we get $\boxed{d_\ell = 4.5}$, while for $\eta \to 0$ we get $\boxed{d_\ell \to 2}$. This small example already shows that for small c we can use a different f to get codelengths much shorter than those obtained by the regular Tardos construction (e.g. $d_\ell > 20$ for $c = 2$).

So one could investigate the use of discrete distributions f for the Tardos scheme, and in particular study the paper by Nuida et al. After investigating this paper and analyzing and understanding their results, one could ask questions like: Can we improve upon their results? Can we tighten their analysis? Can we improve upon their asymptotic result of $\ell \to 5.35c^2 \ln(n/\epsilon_1)$? Can we derive lower bounds for the codelength in the case of such distribution functions F? Note that as the optimal discrete distributions for given c take on c different values, one would expect that for $c \to \infty$ one obtains in the limit a continuous distribution, and probably the optimal asymptotic continuous distribution as well. So why are the asymptotic results described in Chapter 8 better than those in Nuida et al. [NFH+09]?

13.3.2 The Tardos scheme: Bigger alphabets

In Chapter 10 we saw a new, dynamic Tardos scheme to trace a whole coalition using a binary alphabet and a codelength of $\ell = \mathcal{O}(c^2 \ln(n/\epsilon_1))$. The reason of using a binary alphabet the was that the analysis is more simple in the binary case, and that we have clear results for the binary case. So one could investigate whether using non-binary alphabets would lead to significant further improvements. Especially for the new Tardos scheme this is a relevant question, since there because of the offsets one needs to calculate scores per user and per c. With no offset δ one could use the same distribution function for all c, and only maintain one score per user. For this one would first have to see whether it is possible to use a non-binary Tardos scheme with $\delta = 0$ for $c < \infty$, i.e. see if with some different proof method one can prove secureness for this case.

13.3.3 Rate of c-secure frameproof codes

Chapter 4 contains a conjecture (Conjecture 4.36) about the minimal rate of c-secure frameproof codes, which involves intersecting set systems and projective planes over $\mathbb{F}_2$. One obvious problem is: Can we prove or disprove this conjecture? This is a minor problem, and it is more a theoretical question related to intersecting set systems and projective planes than a practical question related to fingerprinting schemes. Maybe one can solve it in a few hours, or maybe this is just a tough problem to crack.

13.3.4 Deterministic dynamic schemes for known coalition sizes

The lower bounds derived by Berkman et al. all need the requirement that c is unknown to the tracer. But what happens when c is known? So (a) what happens to the lower bounds (do the same lower bounds hold?), and (b) can we then get better schemes? For one, we can improve the degree algorithm when c is known, as we immediately lose a factor c in the running time for determining c. But more importantly: how does the optimal algorithm change when c is known? Or can we construct a new algorithm specifically for known c, that is more efficient or simpler than the optimal algorithm?

13.3.5 Probabilistic dynamic schemes: Lower bounds

As the area of probabilistic dynamic schemes is still somewhat unexplored, there are many questions one could ask here. One interesting one may be whether we can prove any good lower bounds on the time (length) for any probabilistic dynamic scheme. Could schemes exist that are orders of magnitude better than for example the universal Tardos scheme?

13.3.6 Other schemes

One could also investigate other schemes not discussed in this report, such as the Blu-ray implementation by Jin and Lotspiech. How do other schemes compare to the schemes discussed here? What are advantages/disadvantages of those schemes?

13.3.7 Watermarking

Related to fingerprinting is watermarking, i.e. the process of actually embedding the codewords into the digital data. One could investigate how the embedding is done in practice, and what this means for the fingerprinting model. Perhaps the real world model is then even different from the models given here, and maybe that model requires a different solution. Or one could investigate the problem of watermarking on itself, which forms a whole new problem of its own.

Appendix A

Optimal symmetric Tardos traitor tracing codes

The following paper, which has not yet been submitted, is a result of the work described in Chapter 8. The work is based partly on work by Blayer and Tassa [BT08], and on work by Skoric et al. [SKC08], and the results improve upon results from these papers two paper as well as upon results from Nuida et al. [NFH$^+$09] Since these three papers were all published in Designs, Codes and Cryptography, we hope that this journal will also accept our paper.

The paper is close to being finished, but it has not yet been submitted.

Des. Codes Cryptogr. manuscript No.
(will be inserted by the editor)

Optimal symmetric Tardos traitor tracing codes

Thijs Laarhoven · **Benne de Weger**

Received: date / Accepted: date

Abstract For the Tardos fingerprinting scheme, we show that by combining the symbol-symmetric accusation function of Škorić et al. with the improved analysis of Blayer and Tassa we get further improvements. Our construction gives codes that are up to 4 times shorter than Blayer and Tassa's, and up to 2 times shorter than the codes from Škorić et al. Asymptotically, we achieve the theoretical optimal codelength for Tardos' distribution function and the symmetric score function. For large coalitions, our codelengths are asymptotically about 4.93% of Tardos' original codelengths, which also improves upon results from Nuida et al.

Keywords Fingerprinting codes · Watermarking · Traitor tracing schemes

Mathematics Subject Classification (2000) 68P30 · 94B60

1 Introduction

Watermarking digital content allows distributors of copyrighted digital data to embed so-called fingerprints into their data in such a way that each copy of the data can be uniquely identified. These watermarks are made in a robust way, so that users cannot change or remove them from the content. If a copy of the data is then illegally distributed to unauthorized users and intercepted by the distributor, he can extract the fingerprint from the copy and find the person whose fingerprinted data was distributed. Actions can then be taken against this user, to prevent further illegal distribution.

To be able to trace the watermarked data back to the user, we need that the embedded fingerprints for each user are different. However, by comparing their differently watermarked copies of the content, multiple malicious users can form a coalition and

T.M.M. Laarhoven · B.M.M. de Weger
Eindhoven University of Technology, Eindhoven, The Netherlands.
This work was done when the first author was with Irdeto, Eindhoven, The Netherlands. The content is mostly based on the first author's Master's thesis.

detect differences in their content. Assuming that besides the watermarks all copies are the same, this allows coalitions to detect part of the watermark. By editing this data, they can then create a forged copy, which contains the same digital content as their original copies, but has a forged fingerprint that cannot be traced back to them directly. Under the marking assumption, which says that colluders can only detect and edit fingerprint positions if their fingerprints do not all match on that position, there are ways to construct fingerprinting schemes such that any forged copy can be traced back to at least one of the colluders. This involves finding a construction for fingerprints for each of the users, and finding a way to trace back forged copies to guilty users.

1.1 Model

Let $U = \{1, \ldots, n\}$ denote the set of the n users that received watermarked content. Here a user corresponds to one watermarked copy of the content, so a person who possesses several differently watermarked copies of the data is assumed to control multiple users. For each user j the distributor generates a fingerprint (also called a codeword), which is usually denoted by $\mathbf{x}_j$. This codeword is a vector of length ℓ (the codelength) of symbols from an alphabet Q of size q. The case $q = 2$ corresponds to the binary alphabet, which is usually taken as $Q = \{0, 1\}$. All fingerprints together form the fingerprinting code $\mathscr{C} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. A common way of representing this code is by putting all codewords as rows in a matrix X according to $X_{ji} = (\mathbf{x}_j)_i$.

After assigning codewords to users and distributing the watermarked copies, a subset $C \subseteq U$ of c users (called colluders or pirates) may form a coalition to create a forged copy. Using some pirate strategy ρ, a function $Q^{\ell \times c} \to Q^{\ell}$, they construct a forged copy, which has some unknown distorted fingerprint $\rho(X) = \mathbf{y}$ called the forgery. For the pirate strategy ρ, we assume that the marking assumption holds, i.e. if for all $j \in C$ the pirates have $(\mathbf{x}_j)_i = \omega$ for some position i and symbol $\omega \in Q$, then the coalition is forced to output $y_i = \omega$. On other positions, we assume that colluders are free to choose any of the symbols from the alphabet.

Finally, after the coalition has created a forged copy, we assume the distributor intercepts it and extracts the forgery $\mathbf{y}$ from the data. He then runs some tracing algorithm σ on the forgery, to get a subset $\sigma(\mathbf{y}) \subseteq U$ of users that are accused. The accusation is said to be successful if no innocent users are accused (i.e. $\sigma(\mathbf{y}) \subseteq C$) and at least one guilty user is accused (i.e. $\sigma(\mathbf{y}) \cap C \neq \emptyset$).

In the setting of probabilistic schemes, the code X and the tracing algorithm σ may depend on some random variables. The events of not accusing any innocent users (soundness) and accusing at least one guilty user (completeness) then also depend on these random variables. Then, instead of demanding that a fingerprinting scheme is always sound and complete, we may demand that the probability of failure is bounded by some small value ε, where the probability is taken over these random variables. This leads to the following definitions of ε_1-soundness and ε_2-completeness.

Definition 1 (Soundness and completeness) Let $C \subseteq U$ be a coalition of size at most c, and let ρ be some pirate strategy employed by this coalition. Then a fingerprinting

scheme (X,σ) is called ε_1-sound if

$$\mathbb{P}[\sigma(\rho(X)) \not\subseteq C] \leq \varepsilon_1.$$

Similarly, a fingerprinting scheme is called ε_2-complete if

$$\mathbb{P}[\sigma(\rho(X)) \cap C = \emptyset] \leq \varepsilon_2.$$

As we will see later, ε_1/n and ε_2 are closely related in the Tardos fingerprinting scheme. Therefore it is convenient to introduce the notation $\eta = \log(\varepsilon_2)/\log(\varepsilon_1/n)$ such that $\varepsilon_2 = (\varepsilon_1/n)^\eta$, which describes how big ε_2 is, compared to ε_1/n. Also, we sometimes simply say a scheme is secure, to denote that it is sound and complete for certain (implicit) parameters ε_1 and ε_2.

1.2 Related work

In [7], Tardos investigated probabilistic binary fingerprinting schemes where small margins of error are allowed. He proved that a codelength of $\ell = \Omega(c^2 \ln(n/\varepsilon_1))$ is necessary to achieve soundness and completeness, while in the same paper he also gave a construction with a codelength of $\ell = 100c^2 \ln(n/\varepsilon_1)$. This construction is often referred to as the Tardos scheme. In [1,3] the lower bound on the codelength was further tightened, to show that one needs $\ell \geq 2\ln(2)c^2 \ln(n/\varepsilon_1)$ for sufficiently large c and $q = 2$, to achieve soundness and completeness.

Since the scheme of Tardos had a constant 100 in front of the $c^2 \ln(n/\varepsilon_1)$ in the codelength, many papers focused on constructing a scheme with the same order codelength, but with a smaller constant. For example, using a discrete distribution function in the Tardos scheme, Nuida et al. showed in [4] that one can achieve codelengths of $\ell < 5c^2 \ln(n/\varepsilon_1)$ in some cases with small c, while for large c they achieved an asymptotic codelength of $\ell \approx 5.35c^2 \ln(n/\varepsilon_1)$. Using a different approach, Amiri and Tardos showed in [1] that with a computation-heavy construction, one can approach the theoretical lower bound of $\ell = 2\ln(2)c^2 \ln(n/\varepsilon_1)$ for large c.

In this paper we will focus on the binary Tardos scheme, which was introduced in [7] and further analyzed and improved in e.g. [2,4–6]. We will focus on two improvements in particular. In [2], Blayer and Tassa made the proofs of soundness and completeness of [7] tighter by introducing several auxiliary variables which were to be optimized later, instead of fixing them in advance. In that paper the construction of the Tardos scheme essentially remained the same, but it was shown that a codelength of $\ell = 85c^2 \ln(n/\varepsilon_1)$ is also sufficient to prove security. In [5], Škorić et al. did change the scheme, by making the score function of the Tardos scheme symbol-symmetric. This also lead to shorter codelengths, giving asymptotic codelengths of $\ell = (\pi^2 + o(1))c^2 \ln(n/\varepsilon_1) \approx 9.87c^2 \ln(n/\varepsilon_1)$ for large c, while maintaining soundness and completeness. Furthermore assuming that the scores of innocent users and the joint coalition score are normally distributed, Škorić et al. showed in [5, Section 6] that an asymptotic codelength of $\ell = (\frac{\pi^2}{2} + o(1))c^2 \ln(n/\varepsilon_1)$ is then both sufficient and necessary. Since by the Central Limit Theorem these scores will in fact converge to normal distributions for asymptotically large c, this also provides a lower bound on the codelength, when using the Tardos distribution function and the symmetric score function.

1.3 Contributions and outline

Combining the symbol-symmetric score function from Škorić et al. with Blayer and Tassa's sharp analysis, we will prove ε_1-soundness and ε_2-completeness for all $c \geq 2$ and $\eta \leq 1$ with a codelength of $\ell = 23.79c^2 \ln(n/\varepsilon_1)$. This improves upon the codelength from Blayer and Tassa by a factor more than 3.5, and it improves upon the original Tardos scheme by a factor of more than 4. Furthermore, for bigger c and smaller η the constant in front of the $c^2 \ln(n/\varepsilon_1)$ in ℓ further decreases, easily leading to a factor 10 improvement over the original Tardos scheme and a factor slightly less than 4 improvement over the Blayer and Tassa analysis.

Similar to work of Škorić et al., we also look at the asymptotics of our scheme, and show that for large c, we can prove soundness and completeness for a codelength of $\ell = (\frac{\pi^2}{2} + O(c^{-1/3}))c^2 \ln(n/\varepsilon_1) \approx 4.93c^2 \ln(n/\varepsilon_1)$. This improves upon the asymptotic results from Škorić et al. by a factor 2, and we achieve the asymptotic optimal codelength which Škorić et al. proved to be secure under the added assumption that the distributions of scores are normal distributions. We therefore close the gap of a factor 2 between the best known provably secure codelength and the asymptotic optimal codelength, for Tardos' distribution function and the symmetric score function. These results also improve upon the asymptotic codelengths from Nuida et al., who used different discrete distribution functions F, by more than 7%.

The paper is organized as follows. In Section 2 we first give the construction of the (symmetric) Tardos scheme, and compare our results with earlier results from literature. In Sections 3 and 4 we then prove that the soundness and completeness properties hold under our assumptions on the parameters. In Section 5 we then give results similar to those in [2, Section 2.4.5] on how to find the optimal set of parameters that satisfies the conditions for our proof method to work, and minimizes the codelength. There we also give such minimal codelengths, for several values of c and η. Finally in Section 6 we prove the results stated above for asymptotically large c.

2 Construction and results

First we present the construction of the Tardos fingerprinting scheme, as in [2], where we use auxiliary variables d_ℓ, d_z, d_δ for the codelength ℓ, accusation offset Z and cutoff parameter δ respectively. The only difference between our construction and that of Blayer and Tassa is in the score function we use. While Blayer and Tassa used the asymmetric score function from Tardos' original scheme, we use the symbol-symmetric score function from Škorić et al.

2.1 The Tardos fingerprinting scheme

Let $n \geq c \geq 2$ be positive integers, and let $\varepsilon_1, \varepsilon_2 \in (0,1)$ be the desired upper bounds for the soundness and completeness error probabilities respectively. Let us write $k = \ln(n/\varepsilon_1)$ so that $e^{-k} = \varepsilon_1/n$. Let d_ℓ, d_z, d_δ be positive constants, with $d_\delta > 1$. Then the symmetric Tardos fingerprinting scheme works as follows.

1. **Initialization**
   (a) Take the codelength as $\ell = d_\ell c^2 k$. [1]
   (b) Take the accusation offset parameter as $Z = d_z ck$.
   (c) Take the cutoff parameter as $\delta = 1/(d_\delta c)$, and compute $\delta' = \arcsin(\sqrt{\delta})$ such that $0 < \delta' < \pi/4$.
   (d) For each fingerprint position $1 \le i \le \ell$, select $p_i \in [\delta, 1-\delta]$ independently from the distribution defined by the following CDF $F(p)$ and PDF $f(p)$:

$$F(p) = \frac{2\arcsin(\sqrt{p}) - 2\delta'}{\pi - 4\delta'}, \quad f(p) = \frac{1}{(\pi - 4\delta')\sqrt{p(1-p)}}. \tag{1}$$

   The function $f(p)$ is biased towards δ and $1-\delta$ and symmetric around $1/2$.
2. **Codeword generation**
   (a) For each position $1 \le i \le \ell$ and for each user $1 \le j \le n$, select the ith entry of the codeword of user j according to $\mathbb{P}[X_{ji} = 1] = p_i$ and $\mathbb{P}[X_{ji} = 0] = 1 - p_i$.
3. **Accusation**
   (a) For each position $1 \le i \le \ell$ and for each user $1 \le j \le n$, calculate the score S_{ji} according to:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ -\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 0, \\ +\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 0. \end{cases} \tag{2}$$

   (b) For each user $1 \le j \le n$, calculate the total accusation sum $S_j = \sum_{i=1}^{\ell} S_{ji}$. User j is accused if and only if $S_j > Z$.

Under certain conditions on the parameters d_ℓ, d_z, d_δ, which are specified in Subsections 2.2 and 2.3, one can prove soundness and completeness, using (a modified version of) Tardos' proof construction. Note that, since this proof method uses several non-tight bounds, it is very well possible that there exist sets of parameters that do not satisfy these conditions, but still guarantee soundness and completeness. So if the conditions are not satisfied, we can only conclude that the proof method does not work in that case.

2.2 Results for the asymmetric Tardos scheme

In the original Tardos scheme, and in several papers discussing the Tardos scheme, the score function is asymmetric in y_i, as only the positions with $y_i = 1$ are taken into account for the accusations. The construction of this asymmetric Tardos scheme is the same as in Subsection 2.1, but with the scores from (2) replaced by:

$$S_{ji} = \begin{cases} +\sqrt{(1-p_i)/p_i} & \text{if } X_{ji} = 1, y_i = 1, \\ -\sqrt{p_i/(1-p_i)} & \text{if } X_{ji} = 0, y_i = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

[1] Note that ℓ may not be integral, while the codelength of a code of course has to be integral. See Appendix A for a short note on how to solve this minor problem in our construction.

Blayer and Tassa performed an extensive analysis of this scheme in [2], and showed that under the following assumptions, one can prove soundness and completeness for given c and η. In these Theorems, the function $h : (0,\infty) \to (\frac{1}{2},\infty)$ is defined by $h(x) = \lambda$ if and only if $e^x = 1 + x + \lambda x^2$. The function $h^{-1} : (\frac{1}{2},\infty) \to (0,\infty)$ denotes its inverse function, and is defined by $h^{-1}(x) = (e^x - 1 - x)/x^2$.

Theorem 1 *[2, Theorem 1.1] Let the Tardos scheme be constructed as in Subsection 2.1, with the asymmetric score function from* (3)*. Let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the following two requirements:*

$$d_\alpha \geq \frac{\sqrt{d_\delta}}{h(r)\sqrt{c}}, \tag{S1}$$

$$\frac{d_z}{d_\alpha} - \frac{r d_\ell}{d_\alpha^2} \geq 1. \tag{S2}$$

Then the scheme is ε_1-sound.

Theorem 2 *[2, Theorem 1.2] Let the Tardos scheme be constructed as in Subsection 2.1, with the asymmetric score function from* (3)*. Let s, g be positive constants such that d_ℓ, d_z, d_δ, s and g satisfy the following two requirements:*

$$\frac{1 - \frac{2}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g, \tag{C1}$$

$$g d_\ell - d_z \geq \eta \sqrt{\frac{d_\delta}{s^2 c}}. \tag{C2}$$

Then the scheme is ε_2-complete.

Tardos' original choice of parameters was the following, which allowed him to prove his scheme is ε_1-sound and ε_2-complete for all $c \geq 2$ and $\eta \leq \sqrt{c}/4$ [7, Theorems 1 and 2]:

$$d_\ell = 100,\ \ d_z = 20,\ \ d_\delta = 300,\ \ d_\alpha = 10,\ \ r = 1,\ \ s = 1,\ \ g = \frac{1}{4}.$$

Blayer and Tassa proved that to achieve ε_1-soundness and ε_2-completeness for all $c \geq 2$ and $\eta \leq 1$, the following choice of parameters is also provably secure [2, Section 2.4]:

$$d_\ell = 85,\ \ d_z = 15,\ \ d_\delta = 40,\ \ d_\alpha = 8,\ \ r = 0.611,\ \ s = 0.757,\ \ g = 0.2461.$$

In [6, Corollary 1], Škorić et al. showed that the following choice of parameters suffices to prove soundness and completeness for asymptotically large c:

$$d_\ell \to 4\pi^2,\ \ d_z \to 4\pi,\ \ d_\delta \to \infty,\ \ d_\alpha \to 2\pi,\ \ r = 1,\ \ s = h(1),\ \ g \to \frac{1}{\pi}.$$

According to the Central Limit Theorem, the scores of innocent users and the total score of the coalition converge to certain normal distributions. Under the assumption that the scores behave exactly like these normal distributions, Škorić et al. showed in

[6, Corollary 3] that the following choice of parameters is then sufficient and necessary to prove soundness and completeness:

$$d_\ell \to 2\pi^2, \;\; d_z \to 2\pi, \;\; d_\delta \to \infty.$$

Applying the analysis from Section 6 to the asymmetric Tardos scheme, we can prove that the following choice of parameters is provably sufficient for large c:[2]

$$d_\ell \to 2\pi^2, \;\; d_z \to 2\pi, \;\; d_\delta \to \infty, \;\; d_\alpha \to \pi, \;\; r \to \frac{1}{2}, \;\; s \to \infty, \;\; g \to \frac{1}{\pi}.$$

So with Blayer and Tassa's proof construction, we obtain a 2 times shorter asymptotic codelength compared to the shortest provable codelength of Škorić et al. for the asymmetric Tardos scheme, and we achieve the asymptotic optimal codelength for the asymmetric Tardos scheme which Škorić et al. only achieved when they added the assumption that scores behave like normal distributions.

2.3 Results for the symmetric Tardos scheme

We will prove in Sections 3 and 4 that with the following assumptions on the parameters, we can also prove soundness and completeness for the symmetric Tardos scheme.

Theorem 3 *Let the Tardos scheme be constructed as in Subsection 2.1, and let d_α, r be positive constants, with $r > \frac{1}{2}$, such that $d_\ell, d_z, d_\delta, d_\alpha$ and r satisfy the requirements from* (S1) *and* (S2)*. Then the scheme is ε_1-sound.*

Theorem 4 *Let the Tardos scheme be constructed as in Subsection 2.1, and let s, g be positive constants, such that d_ℓ, d_z, d_δ, s and g satisfy* (C2) *and the following requirement:*

$$\frac{2 - \frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g. \qquad \text{(C1')}$$

Then the scheme is ε_2-complete.

Using the above results, in Section 5 we will prove ε_1-soundness and ε_2-completeness for all $c \geq 2$ and $\eta \leq 1$ for the following set of parameters:

$$d_\ell = 23.79, \;\; d_z = 8.06, \;\; d_\delta = 28.31, \;\; d_\alpha = 4.58, \;\; r = 0.67, \;\; s = 1.07, \;\; g = 0.49.$$

This improves upon the constants from Blayer and Tassa by a factor more than 3.5, and it improves upon the original Tardos scheme by a factor more than 4. Furthermore, for bigger c and smaller η the values of d_ℓ further decrease, easily leading to a factor 10 improvement over the original Tardos scheme.

[2] These results can be obtained by applying the analysis from Section 6 to Blayer and Tassa's original analysis for the asymmetric Tardos scheme. The main difference is that then one needs $g = \frac{1}{\pi} + o(1)$ instead of $g = \frac{2}{\pi} + o(1)$, which causes an extra factor 4 for d_ℓ and extra factors 2 for d_z and d_α.

Škorić et al. showed that for asymptotically large c, the following set of parameters is sufficient for proving soundness and completeness in the symmetric Tardos scheme [5, Corollary 1]:

$$d_\ell \to \pi^2,\ \ d_z \to 2\pi,\ \ d_\delta \to \infty,\ \ d_\alpha \to \pi,\ \ r = 1,\ \ s = h(1),\ \ g \to \frac{2}{\pi}.$$

With the added assumption that the scores of innocent users and the joint score of guilty users are normally distributed, Škorić et al. also showed that the following set of parameters is sufficient for soundness and completeness, for asymptotically large c [5, Corollary 2]:

$$d_\ell \to \frac{\pi^2}{2},\ \ d_z \to \pi,\ \ d_\delta \to \infty.$$

Since by the Central Limit Theorem these scores will also converge to normal distributions, this shows that the asymptotic optimal codelength for the symmetric Tardos scheme is $\ell = (\frac{\pi^2}{2} + o(1))c^2 \ln(n/\varepsilon_1)$. We show in Section 6 that for asymptotically large c, we can actually prove soundness and completeness with this asymptotic codelength, without any added assumptions. In the asymptotic case, our construction gives the following parameters:

$$d_\ell \to \frac{\pi^2}{2},\ \ d_z \to \pi,\ \ d_\delta \to \infty,\ \ d_\alpha \to \frac{\pi}{2},\ \ r \to \frac{1}{2},\ \ s \to \infty,\ \ g \to \frac{2}{\pi}.$$

Similar to the asymmetric case, we thus get a factor 2 improvement over Škorić et al.'s best provable asymptotic codelength, and we achieve the asymptotic optimal codelength which Škorić et al. only proved with the added assumption that the scores behave like normal distributions. This also improves upon results from Nuida et al. in [4], who showed that with certain discrete distribution functions F, one can prove secureness for $\ell \approx 5.35c^2 \ln(n/\varepsilon_1)$ for large c. With our construction, we show a codelength of $\ell \approx 4.93c^2 \ln(n/\varepsilon_1)$ is provably secure for large c.

3 Soundness

Here we will prove Theorem 3, i.e. prove the soundness property from Definition 1, under the assumptions (S1) and (S2). We will closely follow the proof of soundness of Blayer and Tassa of [2, Theorem 1.1]. We will first prove an upperbound on $\mathbb{E}\left[e^{\alpha S_j}\right]$, with $\alpha = 1/(d_\alpha c)$, and then use this result to prove upper bounds on $\mathbb{P}[j \in \sigma(\mathbf{y})]$ for innocent users j, and $\mathbb{P}[\sigma(\rho(X)) \not\subseteq C]$.

Lemma 1 *Let d_α and r be positive constants, with $r > \frac{1}{2}$, such that d_δ, d_α and r satisfy Equation* (S1)*. Let j be an innocent user, and let S_j be the user's score in the Tardos scheme from Subsection 2.1. Let $\alpha = 1/(d_\alpha c)$. Then*

$$\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{\alpha S_j}\right] \le e^{-r\alpha^2\ell}. \tag{4}$$

Proof First we fill in $S_j = \sum_{i=1}^{\ell} S_{ji}$ and use that S_j does not depend on $X_{j'i}$ for $j' \neq j$ to get

$$\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{\alpha S_j}\right] = \mathbb{E}_{\mathbf{y},\mathbf{X}_j,\mathbf{p}}\left[\prod_{i=1}^{\ell} e^{\alpha S_{ji}}\right] = \prod_{i=1}^{\ell} \mathbb{E}_{y_i,X_{ji},p_i}\left[e^{\alpha S_{ji}}\right].$$

Since $S_{ji} < \sqrt{1/\delta} = \sqrt{d_\delta c}$ it follows that $\alpha S_{ji} < \sqrt{d_\delta}/(d_\alpha\sqrt{c})$. From (S1) we know that $\sqrt{d_\delta}/(d_\alpha\sqrt{c}) \leq h(r)$ for our choice of r, hence $\alpha S_{ji} < h(r)$. From the definition of h we know that $e^w \leq 1+w+rw^2$ exactly when $w \leq h(r)$. Using this with $w = \alpha S_{ji}$ we get

$$\mathbb{E}\left[e^{\alpha S_{ji}}\right] \leq \mathbb{E}\left[1+\alpha S_{ji}+r(\alpha S_{ji})^2\right] = 1+\alpha\mathbb{E}[S_{ji}]+r\alpha^2\mathbb{E}[S_{ji}^2].$$

We can easily calculate $\mathbb{E}[S_{ji}]$ and $\mathbb{E}[S_{ji}^2]$, as y_i and X_{ji} are independent for innocent users j. As in [5, Lemmas 2 and 3], we obtain

$$\mathbb{E}[S_{ji}] = 0, \;\; \mathbb{E}[S_{ji}^2] = 1. \tag{5}$$

So it follows that $\mathbb{E}\left[e^{\alpha S_{ji}}\right] \leq 1+r\alpha^2 \leq e^{r\alpha^2}$, and $\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{\alpha S_j}\right] \leq e^{r\alpha^2\ell}$, which was to be proven. □

Proof (Proof of Theorem 3) We prove that the probability of accusing any particular innocent user is at most ε_1/n. Since there are at most n innocent users, the probability of not accusing any innocent users is then at least $(1-\varepsilon_1/n)^n \geq 1-\varepsilon_1$, which then proves the scheme is ε_1-sound.

Since a user is accused if and only if his score S_j exceeds Z, we need to prove that $\mathbb{P}[S_j > Z] \leq \varepsilon_1/n$ for innocent users j. First of all, we write $\alpha = 1/(d_\alpha c)$, and we use the Markov inequality and Equation (4) from Lemma 1 to obtain

$$\mathbb{P}[j \in \sigma(\mathbf{y})] = \mathbb{P}[S_j > Z] = \mathbb{P}\left[e^{\alpha S_j} > e^{\alpha Z}\right] \leq e^{-\alpha Z}\mathbb{E}\left[e^{\alpha S_j}\right] \leq e^{-\alpha Z + r\alpha^2\ell}.$$

Since we want to prove that $\mathbb{P}[j \in \sigma(\mathbf{y})] \leq \varepsilon_1/n$, the proof would be complete if $e^{-\alpha Z+r\alpha^2\ell} \leq e^{-k} \leq \varepsilon_1/n$, i.e. if $-\alpha Z + r\alpha^2\ell \leq -k$. Filling in $\alpha = 1/(d_\alpha c), Z = d_z ck, \ell = d_\ell c^2 k$, and dividing both sides by $-k$, we get

$$\frac{d_z}{d_\alpha} - \frac{rd_\ell}{d_\alpha^2} \geq 1.$$

This is exactly inequality (S2), which was assumed to hold. This completes the proof. □

Compared to the original proof in [2], this proof has barely changed. The only difference is that now the scores are counted for all positions i, instead of only those positions where $y_i = 1$. However, since in the proof in [2] this number of positions was bounded by ℓ, the result remains the same. This explains why we can prove ε_1-soundness with the symmetric score function under the same assumptions (S1), (S2) as in [2].

4 Completeness

For the proof of Theorem 4, we will again closely follow the proof of Blayer and Tassa of [2, Theorem 1.2], and make changes where necessary to incorporate the symbol-symmetric score function. We first give a Lemma to bound the expectation value of $\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{-\beta S}\right]$ with $\beta = s\sqrt{\delta}/c$ and $S = \sum_{j\in C} S_j$, and then use this Lemma to prove completeness.

Lemma 2 *Let s and g be positive constants such that d_δ, s and g satisfy* (C1'). *Let $\beta = s\sqrt{\delta}/c$, let C be a coalition of size c, and let $S = \sum_{j\in C} S_j$ be their total coalition score in the Tardos scheme from Subsection 2.1. Then*

$$\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{-\beta S}\right] \leq e^{-g\beta\ell}. \tag{6}$$

The proof of Lemma 2 is quite lengthy and can be found in Appendix B. Using this Lemma we can easily prove Theorem 4.

Proof (Proof of Theorem 4) We will prove that for a coalition of size c, with probability at least $1-\varepsilon_2$ the algorithm will accuse at least one of the colluders. Since a coalition of size c can simulate a smaller coalition by disregarding some of its users, this then proves that for any coalition of size at most c, the scheme will accuse at least one of the colluders with probability at least $1-\varepsilon_2$. Note that if no colluders are accused, then the score of each colluder is below Z. Hence if the total coalition score S exceeds cZ, then at least one of the pirates is accused. So to prove ε_2-soundness, it suffices to prove that $\mathbb{P}[S < cZ] \leq \varepsilon_2$.

We first use the Markov inequality and Lemma 2 with $\beta = s\sqrt{\delta}/c > 0$ to get

$$\mathbb{P}[\sigma(\mathbf{y}) \cap C = \emptyset] \leq \mathbb{P}[S < cZ] = \mathbb{P}\left[e^{-\beta S} > e^{-\beta cZ}\right] \leq e^{\beta cZ}\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{-\beta S}\right] \leq e^{\beta cZ - g\beta\ell}.$$

Since we want to prove that $\mathbb{P}[S < cZ] \leq e^{-\eta k} \leq (\varepsilon_1/n)^\eta = \varepsilon_2$, the proof would be complete if $\beta cZ - g\beta\ell \leq -\eta k$. Filling in $\beta = s\sqrt{\delta}/c, \ell = d_\ell c^2 k, Z = d_z ck, \delta = 1/(d_\delta c)$ and writing out both sides, we get

$$g d_\ell - d_z \geq \eta\sqrt{\frac{d_\delta}{s^2 c}}.$$

This is exactly inequality (C2), which was assumed to hold. This completes the proof. □

Compared to [2], we see that instead of using (C1), we now need that inequality (C1') holds. Comparing these two inequalities, we see that a term $\frac{1}{\pi}$ has changed to a $\frac{2}{\pi}$, and a term $\frac{2}{d_\delta \pi}$ has changed to a $\frac{4}{d_\delta \pi}$. The most important change is the $\frac{1}{\pi}$ changing to a $\frac{2}{\pi}$, since that term is the most dominant factor (and the only positive term) on the left hand side of (C1'). By increasing this by a factor 2, we get that $g \leq \frac{2}{\pi}$ instead of $g \leq \frac{1}{\pi}$. Especially for large c, this will play an important role, and it will basically be the reason why the required codelength can then be reduced by a factor 4, compared to Blayer and Tassa's analysis for the asymmetric scheme.

While the other change (the $\frac{2}{d_\delta \pi}$ changing to $\frac{4}{d_\delta \pi}$) does not have a big impact on the optimal choice of parameters for large c, this change does influence the required codelength for smaller c. Because of this change, we now subtract more from the left hand side of (C1'), so that the value of g is bounded sharper from above. This means that for finite values of c, we cannot reduce the codelength of Blayer and Tassa by a factor 4, but only by a factor slightly less than 4.

Finally, after using (C1') in the proof above, the analysis remained the same as in [2]. So under the same assumption (C2) as in [2], we could also complete the proof for the symmetric Tardos scheme.

5 Optimization

Similar to the analysis done by Blayer and Tassa in [2, Section 2.4], we also investigate the optimal choice of parameters such that all requirements are satisfied, and d_ℓ is minimized. As only one of the inequalities has changed, and it changed only on two positions, the formulas for the optimal values of $d_\delta, d_\alpha, d_z, d_\ell$ in the following Theorem are almost the same as in [2, Section 2.4.5].

Theorem 5 *Let η, c be given, and let r,s,g be fixed, satisfying $r \in (\frac{1}{2},\infty), s \in (0,\infty), g \in (0,\frac{2}{\pi})$. Then the optimal choice of $d_\delta, d_\alpha, d_z, d_\ell$, minimizing d_ℓ and satisfying conditions* (S1),(S2),(C1'),(C2)*, is given by:*

$$\hat{d_\delta} = \left(\frac{1}{\frac{4}{\pi} - 2g} \left(\sqrt{\frac{(h^{-1}(s)s)^2}{c} + \frac{16}{\pi}\left(\frac{2}{\pi} - g\right)} + \frac{h^{-1}(s)s}{\sqrt{c}} \right) \right)^2, \tag{O1}$$

$$\hat{d_\alpha} = \max\left(\frac{\sqrt{\hat{d_\delta}}}{h(r)\sqrt{c}}, \frac{r}{g} + \sqrt{\left(\frac{r}{g}\right)^2 + \frac{r}{g}\eta\sqrt{\frac{\hat{d_\delta}}{s^2 c}}} \right), \tag{O2}$$

$$\hat{d_z} = \frac{g\hat{d_\alpha}^2 + r\eta\sqrt{\frac{\hat{d_\delta}}{s^2 c}}}{g\hat{d_\alpha} - r}, \tag{O3}$$

$$\hat{d_\ell} = \frac{\eta\sqrt{\frac{\hat{d_\delta}}{s^2 c}} + \hat{d_z}}{g}. \tag{O4}$$

So to find the optimal septuple $(\hat{r},\hat{s},\hat{g},\hat{d_\delta},\hat{d_\alpha},\hat{d_z},\hat{d_\ell})$ for given c,η, satisfying all requirements and minimizing $\hat{d_\ell}$, one only has to find the triple (r,s,g) with $r \in (\frac{1}{2},\infty), s \in (0,\infty)$ and $g \in (0,\frac{2}{\pi})$ that minimizes the right hand side of (O4).

Example An optimal solution to (S1),(S2),(C1'),(C2) for $c \geq 2$ and $\eta = 1$, minimizing d_ℓ, is given by

$$d_\ell = 23.79,\ d_z = 8.06,\ d_\delta = 28.31,\ d_\alpha = 4.58,\ r = 0.67,\ s = 1.07,\ g = 0.49.$$

This means that with these constants, we can prove soundness and completeness for all $c \geq 2$ and $\eta \leq 1$, with a codelength of $\ell = 23.79c^2\ln(n/\varepsilon_1)$. Compared to the

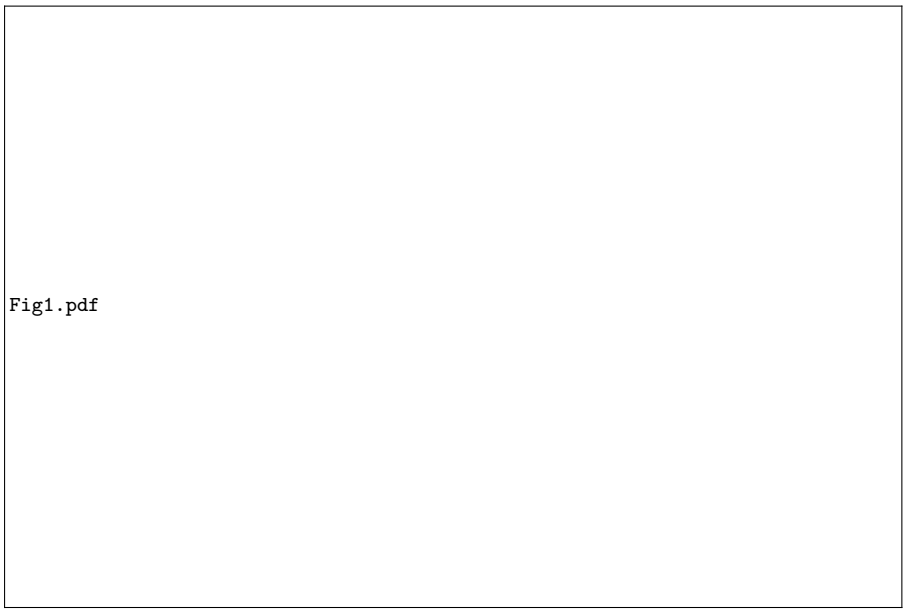

Fig. 1 Optimal values of d_ℓ, for several values of c between 2 and 1000. The different lines correspond to the cases $\eta = 1, 0.5, 0.2, 0.1, 0.01$ respectively, where higher values of η correspond to higher values of $\hat{d_\ell}$.

original Tardos scheme, which had a codelength of $\ell = 100c^2 \lceil \ln(n/\varepsilon_1) \rceil$, this gives an improvement of a factor more than 4. Furthermore we can prove that this scheme is ε_1-sound and ε_2-complete for any value of $c \geq 2$ and $\eta \leq 1$, while Tardos' original proof only works for $c \geq 2$ and $\eta \leq \sqrt{c}/4$.

Example In practice, one usually has $\eta \ll 1$ instead of $\eta = 1$. For example, it could be that $\varepsilon_2 = 1/2$ is sufficient, while $\varepsilon_1 = 10^{-3}$ is desired and there are $n = 10^6$ users, so that $\eta \approx 0.033$. Then the optimizations give us $d_\ell \approx 10.89$ for $c = 2$. So with this larger value of ε_2, a codelength of $\ell = 10.89c^2 \ln(n/\varepsilon_1)$ is sufficient to prove the soundness and completeness properties for any $c \geq 2$. This is then already a factor more than 9 improvement compared to the original Tardos scheme.

If we let c increase in inequalities (O1),(O2),(O3),(O4), i.e. if we only want provable security for $c \geq c_0$ for some $c_0 > 2$, then one can easily see that the inequalities become weaker and an even shorter codelength can be achieved. Figure 1 shows the optimal values of d_ℓ against different values of c, for several values of η. One can see that for large c, a codelength of $\ell < 6c^2 \ln(n/\varepsilon_1)$ can be sufficient. In the next Section, we will see that for large c, the optimal values of d_ℓ will converge to $\frac{\pi^2}{2} \approx 4.93$.

6 Asymptotics

Here we show that with the symmetric Tardos construction, for $c \to \infty$ we can prove secureness for $d_\ell = \frac{\pi^2}{2} + O\left(c^{-1/3}\right)$. We calculate the optimal first order error term

explicitly, and also show explicitly the dependence on η, as the choice of η may depend on the particular application. Note that at least $\eta \leq 1$, but it may be considerably smaller and it may depend on c as well.

Theorem 6 *Let $\gamma = \left(\frac{2}{3\pi}\right)^{2/3} \approx 0.35577$. The optimal asymptotic (for $c \to \infty$) value for d_ℓ, and the accompanying values for d_z, d_δ, are*

$$d_\ell = \frac{\pi^2}{2}\left(1 + \left(3\gamma + 18\gamma\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right), \tag{7}$$

$$d_z = \pi\left(1 + \left(\frac{5}{2}\gamma + 6\gamma\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right), \tag{8}$$

$$d_\delta = \frac{4}{\gamma}\left(1 - 3\frac{\eta}{\log c}(1+o(1))\right)c^{1/3}, \tag{9}$$

and the choices for g, r, s leading to them are given by

$$g = \frac{2}{\pi}\left(1 - \left(\frac{1}{2}\gamma + 3\gamma\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right), \tag{10}$$

$$r = \frac{1}{2}\left(1 + \left(2\gamma - 6\gamma\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right), \tag{11}$$

$$s = \log\left(\frac{24}{\pi^2\gamma}\frac{\eta}{\log c}(1+o(1))c^{1/3}\right). \tag{12}$$

We have optimized for d_ℓ, and one could get slightly better error terms for d_z or d_δ. For example, optimizing for d_z yields an optimal value of $\pi\left(1 + \left(3\gamma' + 9\gamma'\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right)$, for a suboptimal d_ℓ of $\frac{\pi^2}{2}\left(1 + \left(4\gamma' + 15\gamma'\frac{\eta}{\log c}(1+o(1))\right)c^{-1/3}\right)$, where $\gamma' = 2^{-1/3}\gamma$.

It is remarkable that the error terms for d_ℓ and d_z scale with $c^{-1/3}$, while Škorić et al. found error terms scaling with $c^{-1/2}$. It turns out that in [5] an error term in $\widetilde{\mu}$ was not taken into account, and if one does do, their analysis for the binary case will also yield error terms scaling with $c^{-1/3}$. Also note that d_δ (related to the cutoff) scales with $c^{1/3}$, i.e. the cutoff $\frac{1}{d_\delta c}$ scales with $c^{-4/3}$ rather than with c^{-1} as one might have guessed.

An immediate consequence of Theorem 6 is the following result, which shows that asymptotically we will achieve codelengths of $\ell \approx 4.93c^2\ln(n/\varepsilon_1)$, i.e. codelengths that are about 4.93% of Tardos' original codelengths.

Corollary 1 *For $c \to \infty$ the above construction gives an ε_1-sound and ε_2-complete scheme with parameters*

$$\ell \to \frac{\pi^2}{2}c^2\ln(n/\varepsilon_1),\ \ Z \to \pi c\ln(n/\varepsilon_1),\ \ \delta \to 0.$$

This proves that our symmetric Tardos construction is asymptotically optimal, since for large c we achieve the optimal codelength of $\ell = (\pi^2 + o(1))c^2\ln(n/\varepsilon_1)$.

Remark In the proof of Theorem 6, we use that r can be taken in the neighborhood of $\frac{1}{2}$ to get the final result, $d_\ell = \frac{\pi^2}{2} + O(c^{-1/3+\gamma})$. In [5] however, no such variable r was used, as it was simply fixed at 1. If they had taken r as a parameter in their analysis and had taken it close to $\frac{1}{2}$ in the asymptotic case, then they would have obtained the same asymptotic results as we did above, but still with different first order terms.

Acknowledgements The authors would like to thank Boris Škorić, Jeroen Doumen and Peter Roelse for useful discussions and valuable comments.

A Integral codelengths

One detail we have not taken care of and which is often "swept under the carpet" in other literature, is that the codelength ℓ by definition has to be integral. In the construction of the Tardos scheme however, we said we take $\ell = d_\ell c^2 \ln(n/\varepsilon_1)$, while $\ln(n/\varepsilon_1)$ and d_ℓ may not be integral. To solve the problem of non-integral codelengths, Tardos rounded up $\ln(n/\varepsilon_1)$ and took $d_\ell = 100$ in his original scheme. Blayer and Tassa also rounded up $\ln(n/\varepsilon_1)$ and took $d_\ell = 85$, presumably also to guarantee that ℓ is integral[3]. However, rounding up d_ℓ and $\ln(n/\varepsilon_1)$ could drastically increase the codelength. For example, suppose $n = 10^6$, $\varepsilon_1 = \varepsilon_2 = 0.01$, and $c = 25$. Then $\eta = 0.25$ and $\ln(n/\varepsilon_1) \approx 18.42$, and numerical optimizations give $d_\ell \approx 8.18$. Without rounding we would get a codelength of $\ell \approx 94155$, while with rounding we get $\ell' = 106875$. So then the codelength ℓ' is more than 13.5% higher than ℓ, only because we rounded up both $\ln(n/\varepsilon_1)$ and d_ℓ.

Instead of rounding up inbetween, rounding up the entire codelength to $\ell' = \lceil d_\ell c^2 \ln(n/\varepsilon_1) \rceil$ makes more sense. The codelength is then increased by less than 1 symbol, so we hardly notice the difference in the codelength. However, the proofs we give in Section 3 and 4 are based on $\ell = d_\ell c^2 \ln(n/\varepsilon_1)$, which corresponds to using $d_\ell = \ell/(c^2 \ln(n/\varepsilon_1))$. If we take $\ell' = \lceil \ell \rceil$, then we get $d'_\ell = \lceil \ell \rceil/(c^2 \ln(n/\varepsilon_1)) > d_\ell$ (for $\ell \notin \mathbb{N}$), so that with the same parameters Z and δ we may not be able to prove security anymore. In particular, equation (S2) might not be satisfied if d_ℓ is increased, since (S2) implies that $4rd_\ell \leq d_z^2$. Increasing the left hand side may violate this bound, if we do not also increase d_z.

The following Theorem takes care of this minor problem, by showing that if we can find a solution to (S1), (S2), (C1'), (C2) with a fractional codelength ℓ, then we can also find a solution to these inequalities with the integral codelength $\lceil \ell \rceil$. In particular, we show which scheme parameters ℓ, Z and δ one could take to achieve this result.

Theorem 7 *Let the Tardos scheme be constructed as in Subsection 2.1, and let $(d_\ell, d_z, d_\delta, d_\alpha, r, s, g)$ be a septuple satisfying conditions* (S1)*,* (S2)*,* (C1')*,* (C2) *giving scheme parameters $\ell_0 = d_\ell c^2 \ln(n/\varepsilon_1), Z_0 = d_z c \ln(n/\varepsilon_1)$ and $\delta_0 = 1/(d_\delta c)$. Then the Tardos scheme from Subsection 2.1 with parameters*

$$\ell = \lceil \ell_0 \rceil, \;\; Z = Z_0 + \frac{g}{c}\left(\lceil \ell_0 \rceil - \ell_0\right), \;\; \delta = \delta_0 \tag{13}$$

is ε_1-sound and ε_2-complete.

Proof Let us write $\omega = d_\ell(\lceil \ell_0 \rceil - \ell_0)/\ell_0$. We prove that if the equations hold for $(d_\ell, d_z, d_\delta, d_\alpha, r, s, g)$, then they also hold for $(d'_\ell, d'_z, d_\delta, d'_\alpha, r, s, g)$, where $d'_\ell = d_\ell + \omega, d'_z = d_z + g\omega, d'_\alpha = (d'_z + \sqrt{(d'_z)^2 - 4rd'_\ell})/2$. Since for this set of parameters we get ℓ, Z and δ as in (13), the result then follows.

First note that since d_δ, s and g did not change, both sides of inequality (C1') remain the same and this inequality is still satisfied. For inequality (C2), note that both sides also remained the same, since $gd'_\ell - d'_z = g(d_\ell + \omega) - (d_z + g\omega) = gd_\ell - d_z$. For (S2), we rewrite this inequality as a quadratic inequality in d_α:

$$d_\alpha^2 + (-d_z)d_\alpha + rd_\ell \leq 0. \tag{14}$$

[3] Numerical optimizations show that even a parameter set with $d_\ell \approx 81.25$ exists that satisfies all requirements of Blayer and Tassa.

This inequality is satisfied if d_α lies between the two roots of $d_\alpha^2 + (-d_z)d_\alpha + rd_\ell = 0$, which therefore must exist. These roots exist if and only if $(d'_z)^2 - 4rd'_\ell \geq 0$. Since we know that $d_z^2 - 4rd_\ell \geq 0$ the inequality follows if

$$(d'_z)^2 - 4rd'_\ell = (d_z^2 - 4rd_\ell) + (2gd_z + g^2\omega^2 - 4r) \geq d_z^2 - 4rd_\ell \geq 0.$$

From (S2) and (C2) we know that $g(d_z^2) \geq g(4rd_\ell) \geq 4rd_z$, i.e. $gd_z \geq 4r$. So it follows that $2gd_z + g^2\omega^2 \geq 4r$, which proves the second inequality. The third inequality then follows from (S2).

Finally for (S1), we prove that $d'_\alpha \geq d_\alpha$, while the right hand side remains the same, so that this inequality is still satisfied. Note that d_α is also at most the rightmost root to (14), so $d'_\alpha - d_\alpha$ is bounded by

$$d'_\alpha - d_\alpha \geq \frac{d'_z + \sqrt{(d'_z)^2 - 4rd'_\ell}}{2} - \frac{d_z + \sqrt{d_z^2 - 4rd_\ell}}{2} \geq \frac{g\omega}{2} \geq 0.$$

Here the second inequality follows from earlier calculations that $(d'_z)^2 - 4rd'_\ell \geq d_z^2 - 4rd_\ell$. So this choice of d'_α is at least as high as d_α, so inequality (S1) is satisfied. This completes the proof. □

B Proof of Lemma 2

For proving Lemma 2 we will again closely follow the analysis of Blayer and Tassa, and make changes where necessary.

First, we write the total accusation sum of all colluders together as follows:

$$S = \sum_{i=1}^{\ell} \sum_{j \in C}^{c} S_{ji} = \sum_{i=1}^{\ell} y_i \left(x_i q_i - \frac{c - x_i}{q_i} \right) + \sum_{i=1}^{\ell} (1 - y_i) \left(\frac{c - x_i}{q_i} - x_i q_i \right).$$

Here x_i is the number of ones on the ith positions of all colluders, y_i is the output symbol of the pirates on position i, and we introduced the notation $q_i = \sqrt{(1-p_i)/p_i}$. Following the analysis from e.g. Blayer and Tassa, and Tardos, but using that $S_i = (1 - y_i)\left(\frac{c-x_i}{q_i} - x_i q_i\right)$ for positions i where $y_i = 0$ (instead of $S_i = 0$, as with the asymmetric score function), we can bound the expectation value by

$$\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x \right)^{\ell}, \tag{15}$$

where

$$M_x = \begin{cases} E_{0,x} & \text{if } x = 0, \\ E_{1,x} & \text{if } x = c, \\ \max(E_{0,x}, E_{1,x}) & \text{otherwise,} \end{cases}$$

and, for some random variable p distributed according to F,

$$E_{0,x} = \mathbb{E}_p \left[p^x (1-p)^{c-x} e^{-\beta\left(\frac{c-x}{q} - xq\right)} \right],$$
$$E_{1,x} = \mathbb{E}_p \left[p^x (1-p)^{c-x} e^{-\beta\left(xq - \frac{c-x}{q}\right)} \right].$$

Now, using $\beta = s\sqrt{\delta}/c$, we bound the exponents in $E_{0,x}$ and $E_{1,x}$ as follows.

$$-s = \frac{-\beta c}{\sqrt{\delta}} \leq -\beta c q \leq -\beta \left(xq - \frac{c-x}{q} \right) \leq \frac{\beta c}{q} \leq \frac{\beta c}{\sqrt{\delta}} = s.$$

So $|\beta(xq-(c-x)/q)| \leq s$ for our choice of β. So we can use the inequality $e^w \leq 1+w+h^{-1}(s)w^2$ which holds for all $w \leq s$, with $w = \pm\beta(xq-(c-x)/q)$, to obtain

$$E_{0,x} \leq \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(1+\beta\left(xq-\frac{c-x}{q}\right)+h^{-1}(s)\beta^2\left(xq-\frac{c-x}{q}\right)^2\right)\right],$$
$$E_{1,x} \leq \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(1-\beta\left(xq-\frac{c-x}{q}\right)+h^{-1}(s)\beta^2\left(xq-\frac{c-x}{q}\right)^2\right)\right].$$

Introducing more notation, this can be rewritten to

$$E_{0,x} \leq F_{0,x}+\beta F_{1,x}+h^{-1}(s)\beta^2 F_{2,x},$$
$$E_{1,x} \leq F_{0,x}-\beta F_{1,x}+h^{-1}(s)\beta^2 F_{2,x},$$

where

$$F_{0,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\right],$$
$$F_{1,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(xq-\frac{c-x}{q}\right)\right],$$
$$F_{2,x} = \mathbb{E}_p\left[p^x(1-p)^{c-x}\left(xq-\frac{c-x}{q}\right)^2\right].$$

We first calculate $F_{1,x}$ explicitly. Writing out the expectation value and using the definition of $f(p)$ from (1), we get

$$F_{1,x} = \frac{1}{\pi-4\delta'}\int_{\delta}^{1-\delta} p^x(1-p)^{c-x}\left(\frac{x}{p}-\frac{c-x}{1-p}\right)dp$$

The primitive of the integrand is given by $I(p) = p^x(1-p)^{c-x}$, so we get

$$F_{1,x} = \frac{I(1-\delta)-I(\delta)}{\pi-4\delta'} = \frac{(1-\delta)^x\delta^{c-x}-\delta^x(1-\delta)^{c-x}}{\pi-4\delta'}. \tag{16}$$

For $0 < x < c$, we bound $F_{1,x}$ from above and below as

$$\frac{-\delta^x(1-\delta)^{c-x}}{\pi-4\delta'} \leq F_{1,x} \leq \frac{(1-\delta)^x\delta^{c-x}}{\pi-4\delta'}.$$

Using these bounds for M_x, with $0 < x < c$, we get

$$M_x \leq F_{0,x}+\beta\frac{\max(\delta^x(1-\delta)^{c-x},(1-\delta)^x\delta^{c-x})}{\pi-4\delta'}+h^{-1}(s)\beta^2 F_{2,x}.$$

Since $\delta < 1-\delta$, the maximum of the two terms is the first term when $0 < x \leq c/2$, and it is the second term when $c/2 < x < c$. Note that this bound is different from the one of Blayer and Tassa, since in their analysis they do not have this maximum over two terms, but just the first of these two terms. We cannot prove the same upper bound as Blayer and Tassa, and therefore our bound for $M_x, 0 < x < c$, is slightly weaker than Blayer and Tassa's.

For the positions where the marking assumption applies, i.e. $x = 0$ and $x = c$, we do not use the bounds on $F_{1,x}$, but use the exact formula from (16) to obtain

$$M_0 \leq F_{0,0}-\beta\frac{(1-\delta)^c-\delta^c}{\pi-4\delta'}+h^{-1}(s)\beta^2 F_{2,0},$$
$$M_c \leq F_{0,c}-\beta\frac{(1-\delta)^c-\delta^c}{\pi-4\delta'}+h^{-1}(s)\beta^2 F_{2,c}.$$

The value of M_c is the same as that of Blayer and Tassa, but whereas Blayer and Tassa had $M_0 = F_0$, we get a lower upper bound on M_0. This is essentially the reason why with the symmetric score function we get shorter codelengths than Blayer and Tassa.

Substituting the bounds on M_x in the summation over M_x from (15) gives us

$$\begin{aligned}\sum_{x=0}^{c}\binom{c}{x}M_x &\leq M_0 + M_c + \sum_{x=1}^{c-1}\binom{c}{x}M_x \\ &\leq \left(F_{0,0} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,0}\right) \\ &+ \left(F_{0,c} - \beta\frac{(1-\delta)^c - \delta^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,c}\right) \\ &+ \sum_{x=1}^{\lfloor c/2\rfloor}\binom{c}{x}\left(F_{0,x} + \beta\frac{\delta^x(1-\delta)^{c-x}}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,x}\right) \\ &+ \sum_{x=\lfloor c/2\rfloor+1}^{c-1}\binom{c}{x}\left(F_{0,x} + \beta\frac{(1-\delta)^x\delta^{c-x}}{\pi - 4\delta'} + h^{-1}(s)\beta^2 F_{2,x}\right).\end{aligned}$$

Gathering all terms with $F_{0,x}$ and $F_{2,x}$, and using the substitution $x' = c - x$ for the summation over $\lceil c/2\rceil - 1$ terms, we get

$$\begin{aligned}\sum_{x=0}^{c}\binom{c}{x}M_x \leq \sum_{x=0}^{c}\binom{c}{x}F_{0,x} &- \beta\frac{2(1-\delta)^c}{\pi - 4\delta'} + h^{-1}(s)\beta^2\sum_{x=0}^{c}\binom{c}{x}F_{2,x} \\ &+ \frac{\beta}{\pi - 4\delta'}\left(\delta^c + \sum_{x=1}^{\lfloor c/2\rfloor}\binom{c}{x}\delta^x(1-\delta)^{c-x}\right) \\ &+ \frac{\beta}{\pi - 4\delta'}\left(\delta^c + \sum_{x'=1}^{\lceil c/2\rceil-1}\binom{c}{x'}\delta^{x'}(1-\delta)^{c-x'}\right). \qquad (17)\end{aligned}$$

For the summation over $F_{2,x}$, let us define a sequence of random variables $\{T_i\}_{i=1}^{c}$ according to $T_i = q$ with probability p and $T_i = -1/q$ with probability $1-p$. Similar to the inequalities from (5), we get that $\mathbb{E}_p[T_i] = 0$ and $\mathbb{E}_p[T_i^2] = 1$. Also, since T_i and T_j are independent for $i \neq j$, we have that $\mathbb{E}_p[T_iT_j] = 0$ for $i \neq j$. Therefore we can write

$$\mathbb{E}_p\left[\left(\sum_{i=1}^{c}T_i\right)^2\right] = \sum_{i=1}^{c}\mathbb{E}_p\left[T_i^2\right] + \sum_{i\neq j}\mathbb{E}_p\left[T_iT_j\right] = c.$$

But writing out the definition of the expected value, we see that the left hand side is actually the same as the summation over $F_{2,x}$, so that we get

$$\mathbb{E}_p\left[\left(\sum_{i=1}^{c}T_i\right)^2\right] = \sum_{x=0}^{c}\binom{c}{x}p^x(1-p)^{c-x}\left(xq - \frac{c-x}{q}\right)^2 = \sum_{x=0}^{c}\binom{c}{x}F_{2,x} = c.$$

Also we trivially have that

$$\sum_{x=0}^{c}\binom{c}{x}F_{0,x} = \sum_{x=0}^{c}\binom{c}{x}\mathbb{E}_p\left[p^x(1-p)^{c-x}\right] = \mathbb{E}_p\left[\sum_{x=0}^{c}\binom{c}{x}p^x(1-p)^{c-x}\right] = 1.$$

For the summation over $\lfloor c/2\rfloor$ terms we use the following upper bound, which then also holds for the summation over $\lceil c/2\rceil - 1$ terms:

$$\delta^c + \sum_{x=1}^{\lfloor c/2\rfloor}\binom{c}{x}\delta^x(1-\delta)^{c-x} \leq \sum_{x=1}^{c}\binom{c}{x}\delta^x(1-\delta)^{c-x} = 1 - (1-\delta)^c \leq \delta c.$$

Note that this first inequality is quite sharp. In most cases $\delta \ll 1-\delta$, so that the summation is dominated by the terms with low values of x. Adding the terms with $\lfloor c/2\rfloor < x < c$ (i.e. terms with high powers of δ) to the summation has an almost negligible effect on the value of the summation.

Now applying the previous results to (17), and using $(1-\delta)^c \geq 1-\delta c$, which holds for all c, we get

$$\sum_{x=0}^{c} \binom{c}{x} M_x \leq 1 - \beta \frac{2-4c\delta}{\pi - 4\delta'} + h^{-1}(s)\beta^2 c.$$

We want to prove that, for some $g > 0$,

$$\sum_{x=0}^{c} \binom{c}{x} M_x \leq 1 - \beta \frac{2-4c\delta}{\pi - 4\delta'} + h^{-1}(s)\beta^2 c \leq 1 - g\beta \leq e^{-g\beta}. \tag{18}$$

Filling in $\beta = s\sqrt{\delta}/c$ and $\delta = 1/(d_\delta c)$ and writing out the second inequality, this leads to the requirement that

$$\frac{2 - \frac{4}{d_\delta}}{\pi} - \frac{h^{-1}(s)s}{\sqrt{d_\delta c}} \geq g.$$

This is exactly inequality (C1'), which is assumed to hold. Combining the results from Equations (18) and (17) gives us

$$\mathbb{E}_{\mathbf{y},X,\mathbf{p}}\left[e^{-\beta S}\right] \leq \left(\sum_{x=0}^{c} \binom{c}{x} M_x\right)^{\ell} \leq e^{-g\beta\ell}.$$

This completes the proof. □

C Proof of Theorem 6

We introduce parameters K_g, K_r, K_s, a priori depending on c, to enable us to write

$$g = \frac{2}{\pi} - K_g c^{-1/3}, \quad h(r) = K_r c^{-1/3}, \quad \frac{1}{sh^{-1}(s)} = K_s c^{-1/3}.$$

Clearly K_g, K_r, K_s are positive, and we will assume that K_g and K_r are $O(1)$ for $c \to \infty$. This assumption will be validated later on. Note that we do not demand this for K_s (and indeed, it will turn out that $K_s \to \infty$).

Note that $r = h^{-1}(K_r c^{-1/3}) = \frac{1}{2} + \frac{1}{6}K_r c^{-1/3} + O\left(c^{-2/3}\right)$, so that, with for convenience $R = \frac{r}{g}$, we have

$$R = \frac{\pi}{4} + \left(\frac{\pi^2}{8}K_g + \frac{\pi}{12}K_r\right)c^{-1/3} + O\left(c^{-2/3}\right). \tag{19}$$

Next, for convenience we put $D = \sqrt{\frac{d_\delta}{c}}$, and then we have from (O1) that $D = D_0 c^{-1/3}$, where

$$D_0 = \frac{1}{2K_g K_s}\left(1 + \sqrt{1 + \frac{16}{\pi}K_g K_s^2}\right).$$

Note that D_0 is a decreasing function of K_s, with limiting value $\frac{2}{\sqrt{\pi}}\frac{1}{\sqrt{K_g}}$.

From (O2) we see that $d_\alpha = \max\left\{\frac{D}{h(r)}, x_0\right\}$, where $x_0 = R + \sqrt{R^2 + RD\frac{\eta}{s}}$. Note that

$$x_0 = 2R + \frac{1}{2}D\frac{\eta}{s} + O\left(c^{-2/3}\right), \tag{20}$$

where we used that $\frac{\eta}{s} = o(1)$. Note that (O3) and (O4) imply $d_\ell = \frac{d_\alpha^2 + d_\alpha D\frac{\eta}{s}}{g(d_\alpha - R)}$, and that for $x > R$ we have $\frac{x^2 + xD\frac{\eta}{s}}{x-R} \geq 2x_0 + D\frac{\eta}{s}$, with equality if and only if $x = x_0$. So minimizing d_ℓ comes down to minimizing $\frac{2x_0 + D\frac{\eta}{s}}{g}$ under the constraint $d_\alpha = x_0$. Then $d_z = 2x_0$, so that by (19) and (20) we get

$$d_z = \pi + Z_0 c^{-1/3} + O(c^{-2/3}), \text{ where } Z_0 = \frac{\pi^2}{2}K_g + \frac{\pi}{3}K_r + D_0\frac{\eta}{s},$$

and for d_ℓ we obtain

$$d_\ell = \frac{\pi^2}{2} + L_0 c^{-1/3} + O(c^{-2/3}), \text{ where } L_0 = \frac{\pi^3}{2}K_g + \frac{\pi^2}{6}K_r + \pi D_0\frac{\eta}{s}. \tag{21}$$

To find the main terms in the optimal values for d_ℓ, d_z, d_δ, for the moment we neglect error terms. The fact that $d_\alpha = x_0$ implies that $\frac{D}{h(r)} \leq x_0$, and this is asymptotically equivalent to $\frac{D_0}{K_r} \leq \frac{\pi}{2}$. This can be expanded into $1 + \sqrt{1 + \frac{16}{\pi}K_g K_s^2} \leq \pi K_g K_r K_s$, and this leads to $(\pi^3 K_g K_r^2 - 16)K_s \geq 2\pi^2 K_r$, which actually is two conditions:

$$K_g K_r^2 > \frac{16}{\pi^3} = 0.51602\ldots, \quad K_s \geq \frac{2\pi^2 K_r}{\pi^3 K_g K_r^2 - 16}. \tag{22}$$

This shows that it is impossible to choose both K_g and K_r close to 0, and that it is certainly possible to choose them $O(1)$ as $c \to \infty$. Note that optimizing $\frac{\eta}{s}$ implies taking s as large as possible, but this means taking K_s as small as possible, which is limited by the above condition. Indeed, in minimizing L_0 we would like to minimize K_g and K_r, leading to growing K_s, while also s preferably keeps growing. We will see that this is possible.

In optimizing L_0, to find the main term we also neglect for the moment the term $\pi D_0\frac{\eta}{s}$, as it also tends to 0. So we optimize $L_0' = \frac{\pi^3}{2}K_g + \frac{\pi^2}{6}K_r$ under the constraint $K_g K_r^2 > \frac{16}{\pi^3}$. The minimal value for L_0' is reached for $K_g \to \frac{\gamma}{\pi} \approx 0.11325, K_r \to 6\gamma = 2.1346$, where $\gamma = \left(\frac{2}{3\pi}\right)^{2/3} \approx 0.35577$ is a convenience constant. In this case $K_g K_r^2 \to \frac{16}{\pi^3}$, so $K_s \to \infty$, and $D_0 \to \frac{2}{\sqrt{\pi}}\frac{1}{\sqrt{K_g}} \to 3\pi\gamma \approx 3.3531$. It follows that $L_0' \to \frac{3\pi^2}{2}\gamma \approx 5.2670$.

Let us next be more careful, and not throw away the term $\pi D_0\frac{\eta}{s}$ and the error terms. L_0 as in (21) is a priori a function of K_g, K_r and s. We can take for K_s its exact optimal value according to (22), viz.

$$K_s = \frac{2\pi^2 K_r}{\pi^3 K_g K_r^2 - 16}, \tag{23}$$

so that $D_0 = \frac{\pi}{2}K_r$. Note that (23) allows us to eliminate from L_0 the variable K_g. This yields

$$L_0 = \frac{\pi^2}{6}\left(1 + 3\frac{\eta}{s}\right)K_r + \pi^2\frac{1}{K_r K_s} + 8\frac{1}{K_r^2}.$$

We now minimize L_0 by setting the partial derivatives w.r.t. s and K_r to 0. Indeed, $\frac{\partial L_0}{\partial K_r} = \frac{\pi^2}{6}\left(1 + 3\frac{\eta}{s}\right) - \pi^2\frac{1}{K_r^2 K_s} - 16\frac{1}{K_r^3}$, and this being 0 implies

$$\frac{\pi^2}{6}\left(1 + 3\frac{\eta}{s}\right)K_r^2 - 16\frac{1}{K_r} = \pi^2\frac{1}{K_s}. \tag{24}$$

Further, by $\frac{1}{K_s^2}\frac{dK_s}{ds} = -\frac{(s-1)e^s+1}{s^2}c^{-1/3}$ we find $\frac{\partial L_0}{\partial s} = -\frac{\pi^2}{2}\frac{\eta}{s^2}K_r + \pi^2\frac{1}{K_r}\frac{(s-1)e^s+1}{s^2}c^{-1/3}$, and this being 0 implies

$$K_r^2 = \frac{2}{\eta}((s-1)e^s+1)c^{-1/3}. \tag{25}$$

From (24) and (25) we eliminate K_r, and thus obtain an equation in s only, viz.

$$\left(1+3\frac{\eta}{s}\right)\frac{1}{\eta^{3/2}}((s-1)e^s+1)^{3/2} - \frac{24\sqrt{2}}{\pi^2}c^{1/2} = 3\frac{1}{\eta^{1/2}}\frac{e^s-1-s}{s}((s-1)e^s+1)^{1/2}.$$

The first term on the left hand side is $\left(\frac{se^s}{\eta}\right)^{3/2}\left(1+O\left(\frac{1}{s}\right)\right)$, and the right hand side is $\frac{3(e^s)^{3/2}}{(s\eta)^{1/2}}\left(1+O\left(\frac{1}{s}\right)\right)$, and as $\eta<1$ and $s\to\infty$ the right hand side clearly is smaller, so vanishes in the $O\left(\frac{1}{s}\right)$. So we find $\left(\frac{se^s}{\eta}\right)^{3/2}\left(1+O\left(\frac{1}{s}\right)\right) = \frac{24\sqrt{2}}{\pi^2}c^{1/2}$, and this yields

$$se^s = \left(\frac{8}{\pi^2\gamma}\eta + O\left(\frac{1}{\log c}\right)\right)c^{1/3}.$$

In turn this implies

$$s = \frac{1}{3}\log c - \log\log c + \log\eta + O(1), \quad \frac{1}{s} = \frac{3}{\log c}\left(1+O\left(\frac{\log\log c}{\log c}\right)\right), \tag{26}$$

and

$$K_s = \frac{\pi^2\gamma}{72\eta}\log^2 c\left(1+O\left(\frac{1}{\log c}\right)\right). \tag{27}$$

Indeed we find that K_s and s both tend to ∞.

To get the proper value for K_r we turn to (24), and introduce θ such that $K_r = 6\gamma+\theta$, so that θ will tend to 0. Then (24) becomes a cubic equation in θ:

$$\theta^3 + 18\gamma\theta^2 + \left(108\gamma^2 - \frac{6}{\left(1+3\frac{\eta}{s}\right)K_s}\right)\theta + \left(\frac{\frac{288}{\pi^2}\frac{\eta}{s}}{1+3\frac{\eta}{s}} - \frac{36\gamma}{\left(1+3\frac{\eta}{s}\right)K_s}\right) = 0. \tag{28}$$

When $s\to\infty$ and $K_s\to\infty$, this ultimately becomes $\theta\left(\theta^2+18\gamma\theta+108\gamma^2\right)=0$, with the quadratic term being positive definite, showing that (28) for finite large s has exactly one real solution, which will be close to 0. For this solution we have, using (26), (27),

$$\left(108\gamma^2 + O\left(\frac{1}{\log^2 c}\right)\right)\theta + O(\theta^2) = -\frac{288}{\pi^2}\frac{\eta}{s}\left(1+O\left(\frac{1}{\log c}\right)\right),$$

hence

$$K_r = 6\gamma\left(1-\frac{\eta}{s}\left(1+O\left(\frac{1}{\log c}\right)\right)\right), \quad K_g = \frac{\gamma}{\pi}\left(1+2\frac{\eta}{s}\left(1+O\left(\frac{1}{\log c}\right)\right)\right).$$

Putting everything together, using (26), we find

$$L_0 = \frac{3}{2}\pi^2\gamma\left(1+6\eta\frac{1}{\log c}(1+o(1))\right), \quad Z_0 = \frac{1}{2}\pi\gamma\left(5+12\eta\frac{1}{\log c}(1+o(1))\right).$$

The result now easily follows. □

References

1. Amiri, E., Tardos, G.: High rate fingerprinting codes and the fingerprinting capacity. Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms pp. 336–345 (2009). URL http://portal.acm.org/citation.cfm?id=1496770.1496808
2. Blayer, O., Tassa, T.: Improved versions of Tardos fingerprinting scheme. Designs, Codes and Cryptography **48**, 79–103 (2008). URL http://dx.doi.org/10.1007/s10623-008-9200-z. 10.1007/s10623-008-9200-z
3. Huang, Y.W., Moulin, P.: Saddle-point solution of the fingerprinting capacity game under the marking assumption. Proceedings of the 2009 IEEE international conference on Symposium on Information Theory **4**, 2256–2260 (2009). URL http://portal.acm.org/citation.cfm?id=1700967.1700985
4. Nuida, K., Fujitsu, S., Hagiwara, M., Kitagawa, T., Watanabe, H., Ogawa, K., Imai, H.: An improvement of discrete Tardos fingerprinting codes. Designs, Codes and Cryptography **52**, 339–362 (2009). URL http://dx.doi.org/10.1007/s10623-009-9285-z. 10.1007/s10623-009-9285-z
5. Skoric, B., Katzenbeisser, S., Celik, M.: Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. Designs, Codes and Cryptography **46**, 137–166 (2008). URL http://dx.doi.org/10.1007/s10623-007-9142-x. 10.1007/s10623-007-9142-x
6. Skoric, B., Vladimirova, T., Celik, M., Talstra, J.: Tardos fingerprinting is better than we thought. Information Theory, IEEE Transactions on **54**(8), 3663 –3676 (2008). DOI 10.1109/TIT.2008.926307
7. Tardos, G.: Optimal probabilistic fingerprint codes. In: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, STOC '03, pp. 116–125. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/780542.780561. URL http://doi.acm.org/10.1145/780542.780561

Bibliography

[AB08] Prasanth Anthapadmanabhan and Alexander Barg. Randomized Frameproof Codes: Fingerprinting Plus Validation Minus Tracing. *CoRR*, abs/0802.3419, 2008.

[ABD06] Prasanth Anthapadmanabhan, Alexander Barg, and Ilya Dumer. On the Fingerprinting Capacity Under the Marking Assumption. *CoRR*, abs/cs/0612073, 2006.

[AFS01] Noga Alon, Eldar Fischer, and Mario Szegedy. Parent-Identifying Codes. *Journal of Combinatorial Theory, Series A*, 95(2):349–359, 2001.

[AS04] Noga Alon and Uri Stav. New Bounds on Parent-Identifying Codes: The Case of Multiple Parents. *Comb. Probab. Comput.*, 13:795–807, 2004.

[AT09] Ehsan Amiri and Gabor Tardos. High rate fingerprinting codes and the fingerprinting capacity. *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 336–345, 2009.

[BBK03] Alexander Barg, George Blakley, and Grigory Kabatiansky. Digital fingerprinting codes: Problem statements, constructions, identification of traitors. *IEEE Trans. Inform. Theory*, 49:852–865, 2003.

[BCE+01] Alexander Barg, Gerard Cohen, Sylvia Encheva, Grigory Kabatiansky, and Gilles Zemor. A hypergraph approach to the identifying parent property: the case of multiple parents. *SIAM J. Disc. Math*, 14:423–431, 2001.

[BEN07] Simon Blackburn, Tuvi Etzion, and Siaw-Lynn Ng. Prolific Codes with the Identifiable Parent Property. *Cryptology ePrint Archive, Report 2007/276*, 2007.

[BEN09] Simon Blackburn, Tuvi Etzion, and Siaw-Lynn Ng. Traceability Codes. *Cryptology ePrint Archive, Report 2009/046*, 2009.

[BF99] Dan Boneh and Matthew Franklin. An Efficient Public Key Traitor Tracing Scheme. *Advances in Cryptology - CRYPTO '99*, 1666:783–783, 1999.

[BK03] Alexander Barg and Grigory Kabatiansky. A Class of IPP Codes with Efficient Identification. *Journal of Complexity*, 20:137–147, 2003.

[Bla03a] Simon Blackburn. An Upper Bound on the Size of a Code with the k-Identifiable Parent Property. *J. Comb. Theory Ser. A*, 102:179–185, 2003.

[Bla03b] Simon Blackburn. Combinatorial schemes for protecting digital content. *Surveys in Combinatorics 2003*, pages 43–78, 2003.

[Bla03c] Simon Blackburn. Frameproof codes. *SIAM Journal on Discrete Mathematics*, 16(3):499–510, 2003.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. *Proceedings of the 15th ACM conference on Computer and communications security*, pages 501–510, 2008.

[BPS00] Omer Berkman, Michal Parnas, and Jiri Sgall. Efficient dynamic traitor tracing. *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 586–595, 2000.

[BS98] Dan Boneh and James Shaw. Collusion-Secure Fingerprinting for Digital Data. *IEEE Transactions on Information Theory*, pages 452–465, 1998.

[BS11] Dion Boesten and Boris Skoric. Asymptotic fingerprinting capacity for non-binary alphabets. *arXiv.org Preprint*, 2011.

[BT08] Oded Blayer and Tamir Tassa. Improved versions of Tardos' fingerprinting scheme. *Des. Codes Cryptography*, 48:79–103, 2008.

[CE00] Gerard Cohen and Sylvia Encheva. Efficient constructions of frameproof codes. *Electronics Letters*, 36(22):1840–1842, 2000.

[CFNP00] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[CKLS96] Ingemar Cox, Joe Kilian, Tom Leighton, and Talal Shamoon. A secure, robust watermark for multimedia. *Information Hiding - Lecture Notes in Computer Science*, 1174:185–206, 1996.

[CXFF09] Ana Charpentier, Fuchon Xie, Caroline Fontaine, and Teddy Furon. Expectation Maximization of Tardos probabilistic fingerprinting code. *Preprint*, 2009.

[EC01] Sylvia Encheva and Gerard Cohen. Some new p-ary two-secure frameproof codes. *Applied Mathematics Letters*, 14(2):177–182, 2001.

[EC02] Sylvia Encheva and Gerard Cohen. Frameproof codes against limited coalitions of pirates. *Theoretical Computer Science*, 273(2):295–304, 2002.

[FGC08] Teddy Furon, Arnaud Guyader, and Frederic Cerou. On the Design and Optimization of Tardos Probabilistic Fingerprinting Codes. *Information Hiding - Lecture Notes in Computer Science*, 5284:341–356, 2008.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. *Proceedings of the 13th annual international cryptology conference on Advances in cryptology*, pages 480–491, 1994.

[FPF09] Teddy Furon and Luis Perez-Freire. EM Decoding of Tardos Traitor Tracing Codes. *Proceedings of the 11th ACM workshop on Multimedia and security*, pages 99–106, 2009.

[FT01] Amos Fiat and Tamir Tassa. Dynamic Traitor Tracing. *Journal of Cryptology*, 14:211–223, 2001.

[HM09a] Yen-Wei Huang and Pierre Moulin. Capacity-achieving Fingerprint Decoding. *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, pages 51–55, 2009.

[HM09b] Yen-Wei Huang and Pierre Moulin. Saddle-point Solution of the Fingerprinting Capacity game under the Marking Assumption. *Proceedings of the 2009 IEEE international conference on Symposium on Information Theory*, 4:2256–2260, 2009.

[HVLLT98] Henk Hollmann, Jack Van Lint, Jean-Paul Linnartz, and Ludo Tolhuizen. On Codes with the Identifiable Parent Property. *Journal of Combinatorial Theory, Series A*, 82(2):121–133, 1998.

[JB07] Hongxia Jin and Mario Blaum. Combinatorial Properties for Traceability Codes using Error Correcting Codes. *IEEE Trans. Inform. Theory*, 53:804–808, 2007.

[JKL09] Pascal Junod, Alexandre Karlov, and Arjen Lenstra. Improving the Boneh-Franklin Traitor Tracing Scheme. *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, pages 88–104, 2009.

[JLN04] Hongxia Jin, Jeffery Lotspiech, and Stefan Nusser. Traitor Tracing for prerecorded and recordable media. *Proceedings of the 4th ACM workshop on Digital rights management*, pages 83–90, 2004.

[Ker10] Andrew Ker. The Square Root Law in Stegosystems with Imperfect Information. *Proc. 12th Information Hiding Workshop - Lecture Notes in Computer Science*, 6387:145–160, 2010.

[KSCS07] Stefan Katzenbeisser, Boris Skoric, Mehmet Celik, and Ahmad-Reza Sadeghi. Combining Tardos Fingerprinting Codes and Fingercasting. *Information Hiding - Lecture Notes in Computer Science*, 4567:294–310, 2007.

[LDW11] Thijs Laarhoven and Benne De Weger. Optimal symmetric Tardos fingerprinting codes. *Preprint*, 2011+.

[NFH+09] Koji Nuida, Satoshi Fujitsu, Manabu Hagiwara, Takashi Kitagawa, Hajime Watanabe, Kazuto Ogawa, and Hideki Imai. An Improvement of Discrete Tardos Fingerprinting Codes. *Designs, Codes and Cryptography*, 52:339–362, 2009.

[NNL01] Dalit Naor, Moni Naor, and Jeffrey Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 41–62, 2001.

[Pat07] Maura Paterson. Sequential and Dynamic Frameproof Codes. *Des. Codes Cryptography*, 42:317–326, 2007.

[PSS03] Chris Peikert, Abhi Shelat, and Adam Smith. Lower Bounds for Collusion-secure Fingerprinting. *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 472–479, 2003.

[Ron05] Tor Roneid. Collusion-Secure Fingerprinting - A Simulation of the Boneh and Shaw Scheme. *Master's Thesis*, 2005.

[Sch03] Hans Georg Schaathun. Fighting Two Pirates. *Proceedings of the 15th International Conference on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 71–78, 2003.

[Sch04] Hans Georg Schaathun. Binary Collusion-secure Codes: Comparison and Improvements. *Reports in Informatics, University of Bergen, Norway*, 2004.

[Sch06] Hans Georg Schaathun. The Boneh-Shaw Fingerprinting Scheme is better than we thought. *IEEE Transactions on Information Forensics and Security*, 1:248–255, 2006.

[Sch08] Hans Georg Schaathun. On the Assumption of Equal Contributions in Fingerprinting. *IEEE Transactions on Information Forensics and Security*, 3:569–572, 2008.

[SDF02] Francesc Sebe and Josep Domingo-Ferrer. Short 3-Secure Fingerprinting Codes for Copyright Protection. *Proceedings of the 7th Australian Conference on Information Security and Privacy*, pages 316–327, 2002.

[SDF03] Francesc Sebe and Josep Domingo-Ferrer. Collusion-Secure and Cost-Effective Detection of Unlawful Multimedia Redistribution. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 33:382–389, 2003.

[SKC08] Boris Skoric, Stefan Katzenbeisser, and Mehmet Celik. Symmetric Tardos Fingerprinting Codes for Arbitrary Alphabet Sizes. *Des. Codes Cryptography*, 46(2):137–166, 2008.

[SKSC09] Boris Skoric, Stefan Katzenbeisser, Hans Georg Schaathun, and Mehmet Celik. Tardos Fingerprinting Codes in the Combined Digit Model. *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, pages 41–45, 2009.

[SNW03] Reihaneh Safavi-Naini and Yejing Wang. Sequential Traitor Tracing. *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 316–332, 2003.

[SS01] Palash Sarkar and Douglas Stinson. Frameproof and IPP Codes. *Progress in Cryptology - INDOCRYPT 2001*, 2247:117–126, 2001.

[SS10] Antonino Simone and Boris Skoric. Accusation Probabilities in Tardos Codes: the Gaussian Approximation is better than we thought. *Cryptology ePrint Archive, Report 2010/472*, 2010.

[SS11] Antonino Simone and Boris Skoric. Asymptotically false-positive-maximizing attack on non-binary Tardos codes. *arXiv.org Preprint*, 2011.

[SSW01] Jessica Staddon, Douglas Stinson, and Ruizhong Wei. Combinatorial Properties of Frameproof and Traceability Codes. *IEEE Transactions on Information Theory*, 47:1042–1049, 2001.

[STW00] Douglas Stinson, Tran Van Trung, and Ruizhong Wei. Secure Frameproof Codes, Key Distribution Patterns, Group Testing Algorithms and Related Structures. *Journal of Statistical Planning and Inference*, 86:595–617, 2000.

[SVCT06] Boris Skoric, Tatiana Vladimirova, Mehmet Celik, and Joop Talstra. Tardos Fingerprinting is better than we thought. *CoRR*, abs/cs/0607131, 2006.

[SW98] Douglas Stinson and Ruizhong Wei. Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes. *SIAM Journal on Discrete Mathematics*, 11(1):41–53, 1998.

[SZ08] Douglas Stinson and Gregory Zaverucha. Some Improved Bounds for Secure Frameproof Codes and Related Separating Hash Families. *IEEE Transactions on Information Theory*, 54:2508–2514, 2008.

[Tar03] Gabor Tardos. Optimal Probabilistic Fingerprint Codes. *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 116–125, 2003.

[Tar09] Gabor Tardos. Tracing Traitors - Fingerprinting Digital Documents. *Workshop at the Eindhoven University of Technology, October 7-9, 2009*, 2009.

[Tar10] Gabor Tardos. Capacity of Collusion Secure Fingerprinting - A Tradeoff between Rate and Efficiency. *Information Hiding - Lecture Notes in Computer Science*, 6387:81–85, 2010.

[Tas05] Tamir Tassa. Low Bandwidth Dynamic Traitor Tracing Schemes. *J. Cryptol.*, 18:167–183, 2005.

[TM05] Tran Van Trung and Sosina Martirosyan. New Constructions for IPP Codes. *Des. Codes Cryptography*, 35:227–239, 2005.

[Tur41] Paul Turan. On an Extremal Problem in Graph Theory. *Matematikai és Fizikai Lapok*, 48:436—452, 1941.

[XMS07] Yu Xiong, Jun Ma, and Hao Shen. On Optimal Codes with w-Identifiable Parent Property. *Des. Codes Cryptography*, 45:65–90, 2007.

[Yac01] Yacov Yacobi. Improved Boneh-Shaw Content Fingerprinting. *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, pages 378–391, 2001.