New Considerations about the Correct Design of Turbo Fingerprinting Codes

Joan Tomàs-Buliart[1], Marcel Fernández[1], and Miguel Soriano[1,2,⋆]

[1] Department of Telematics Engineering. Universitat Politècnica de Catalunya. C/ Jordi Girona 1 i 3. Campus Nord, Mod C3, UPC. 08034 Barcelona. Spain
[2] CTTC: Centre Tecnològic de Telecomunicacions de Catalunya Parc Mediterrani de la Tecnologia (PMT), Av. Canal Olímpic S/N, 08860 - Castelldefels, Barcelona (Spain)
{jtomas,marcel,soriano}@entel.upc.edu

Abstract. Since the introduction of turbo codes in 1993, many new applications for this family of codes have been proposed. One of the latest, in the context of digital fingerprinting, is called turbo fingerprinting codes and was proposed by Zhang *et al.*. The main idea is a new fingerprinting code composed of an outer turbo code and an inner code based on the Boneh-Shaw model. The major contribution of this paper is a new analysis of this new family of codes that shows its drawbacks. These drawbacks must be considered in order to perform a correct design of a turbo fingerprinting scheme otherwise the scheme cannot retrieve the traitor users which is the main goal of digital fingerprinting scheme. Moreover, the identification of these drawbacks allows to discuss an entirely new construction of fingerprinting codes based on turbo codes.

Keywords: digital fingerprinting, collusion security, tracing traitor, turbo code.

1 Introduction

The distribution and playback of digital images and other multimedia products is an easy task due to the digital nature of the content. Achieving satisfactory copyright protection has become a challenging problem for the research community. Encrypting the data only offers protection as long as the data remains encrypted, since once an authorized but fraudulent user decrypts it, nothing stops him from redistributing the data without having to worry about being caught.

The concept of fingerprinting was introduced by Wagner in [1] as a method to protect intellectual property in multimedia contents. The fingerprinting technique consists in making the copies of a digital object unique by embedding a

⋆ This work has been supported partially by the Spanish Research Council (CICYT) Project TSI2005-07293-C02-01 (SECONNET), by the Spanish Ministry of Science and Education with CONSOLIDER CSD2007-00004 (ARES) and by Generalitat de Catalunya with the grant 2005 SGR 01015 to consolidated research groups.

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 501–516, 2008.

different set of marks in each copy. Having unique copies of an object clearly rules out plain redistribution, but still a coalition of dishonest users can collude. A collusion attack consist in comparing the copies of the coalition members and by changing the marks where their copies differ, they create a pirate copy that tries to disguise their identities. Observe that in this situation it is possible for the attackers to frame an innocent user. Thus, the fingerprinting problem consists in finding, for each copy of the object, the right set of marks that help to prevent collusion attacks.

The construction of collusion secure codes was first addressed in [2]. In that paper, Boneh and Shaw obtain $(c > 1)$-secure codes, which are capable of identifying a guilty user in a coalition of at most c users with a probability ϵ of failing to do so. The construction composes an inner binary code with an outer random code. Therefore, the identification algorithm involves the decoding of a random code, that is known to be a NP-hard problem [3]. Moreover, the length of the code is considerably large for small error probabilities and a large number of users.

To reduce the decoding complexity, Barg, Blakley and Kabatiansky in [3] used algebraic-geometric codes to construct fingerprinting codes. In this way, their system reduces the decoding complexity to $O(poly(n))$ for a code length n and only 2 traitors. In [4], Fernandez and Soriano constructed a 2-secure fingerprinting code by concatenating an inner $(2, 2)$-separating codes with an outer IPP code (a code with the Identifiable Parent Property), and also with decoding complexity $O(poly(n))$.

The Collusion Secure Convolutional Fingerprinting Information Codes presented in [5] have shorter information encoding length and achieve optimal traitor searching in scenarios with a large number of buyers. Unfortunately, these codes suffer from an important drawback in the form of false positives, in other words, an innocent user can be tagged as guilty with very high probability. In [6] we analysed in depth the work in [5] and quantified the probability of false positives. Turbo fingerprinting codes analyzed in this paper are presented in [7] by Zhang *et al.* and are based in the same idea that Collusion Secure Convolutional Fingerprinting Information Codes but using turbo codes instead of Convolutional codes. In a practical implementation of these codes, the turbo code must have some restrictions, which the authors did not take into account, to obtain the desired performance. In this paper, the problem of false positives in [7] is discussed.

The paper is organized as follows. In section 2 we provide some definitions on fingerprinting and error correcting codes. Section 3 presents the well known Boneh-Shaw fingerprinting codes and, in section 4, turbo codes are introduced. Section 5 discusses the turbo fingerprinting codes presented by Zhang *et al.* and carefully explains the encoding and decoding mechanisms. In section 6, new considerations about Turbo Fingerprinting Codes (TFC) and the errors that can be produced as a consequence of not taking into account these considerations are explained and justified. In the same way a numerical example of this problem is given. Section 7 proposes two improvements to the performance of the TFC

using the likelihood provided by the turbo decoder. Finally, some conclusions are given in Section 8.

2 Definitions

We begin by defining some concepts that will be needed throughout the paper.

Definition 1. *(Error Correcting Code) A set C of N words of length L over an alphabet of p letters is said to be an $(L, N, D)_p$-Error-Correcting Code or in short, an $(L, N, D)_p$-ECC, if the Hamming distance*[1] *between every pair of words in C is at least D.*

Definition 2. *(Codebook [2]) A set $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\} \subseteq \Sigma^l$, where Σ will denote some alphabet of size s, will be called an (l, n)-code. The codeword $w^{(u_i)}$ will be assigned to user u_i, for $1 \leq i \leq n$. We refer to the set of words in Γ as the* ***codebook***

Definition 3. *(Undetectable Position) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n)-code and $C = \{u_1, u_2, \cdots, u_c\}$ be a coalition of c-traitors. Let position i be* ***undetectable*** *for C, i.e. the words assigned to users in C match in i'th position, that is $w_i^{(u_1)} = \cdots = w_i^{(u_c)}$.*

Definition 4. *(Feasible set) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n)-code and $C = \{u_1, u_2, \cdots, u_c\}$ be a coalition of c-traitors. We define the* ***feasible set*** *Γ of C as*

$$\Gamma(C) = \{x = (x_1, \cdots, x_l) \in \Sigma^l \mid x_j \in w_j, 1 \leq j \leq l\}$$

where

$$w_j = \begin{cases} \{w_j^{(u_1)}\} & w_j^{(u_1)} = \cdots = w_j^{(u_c)} \\ \{w_j^{(u_i)} \mid 1 \leq i \leq c\} \cup \{?\} & otherwise \end{cases}$$

where ? denotes an erased position.

Now we are in position to define the *Marking Assumption* that establishes the rules that the attacking coalition is subjected to. This definition sets the work environment of many of the actual fingerprinting schemes.

Definition 5. *(Marking Assumption) Let $\Gamma = \{w^{(1)}, w^{(2)}, \cdots, w^{(n)}\}$ be an (l,n) -code, $C = \{u_1, u_2, \cdots, u_c\}$ a coalition of c-traitors and $\Gamma(C)$ the feasible set of C. The coalition C is only capable of creating an object whose fingerprinting lies in $\Gamma(C)$.*

The main idea of this definition is that a coalition of c-traitors can not detect the positions in the document in which their marks hold the same value. Many of the fingerprinting schemes in the literature base their tracing algorithms in trying to estimate the positions that are changed by the attackers.

[1] The Hamming distance $d_H(y, x)$ [8] between two sequences of equal length can be defined as the number of positions in which the two sequences differ.

3 Boneh-Shaw Fingerprinting Model

In 1995, Dan Boneh and James Shaw presented in [2] a seminal paper about the collusion secure fingerprinting problem. First of all, we need to define what a fingerprinting scheme is.

Definition 6. *(Fingerprinting scheme [2])A (l, n)-fingerprinting scheme is a function $\Gamma(u, r)$ which maps a user identifier $1 \geq u \geq n$ and a string of random bits $r \in \{0, 1\}^*$ to a codeword Σ^l. The random string r is the set of random bits used by the distributor and kept hidden from the user. We denote a fingerprinting scheme by Γ_r.*

3.1 *n*-Secure Codes

We now define n-secure codes, see [2] for a more detailed description.

Definition 7. *A fingerprinting scheme Γ_r is a c-secure code with ϵ-error if there exists a tracing algorithm A which from a word x, that has been generated (under the Marking Assumption) by a coalition C of at most c users, satisfies the following condition $Pr[A(x) \in C] > 1 - \epsilon$ where the probability is taken over random choices made by the coalition.*

Now, we define the code and its decoding algorithm:

1. Construct an n-secure (l, n)-code with length $l = n^{O(1)}$.
2. Construct an $\Gamma_0(n, d)$-fingerprinting scheme by replicating each column of an (l, n)-code d times. For example, suppose a $(3, 4)$-code {111,011,001,000}. We can construct a $\Gamma_0(4, 3)$ for four users A,B,C and D as follows:

$$\begin{aligned} A &: 111111111 \\ B &: 000111111 \\ C &: 000000111 \\ D &: 000000000 \end{aligned}$$

3. When the code has been defined, the next step is to define the appropriate decoding algorithm. For instance

   Algorithm 1. *From [2], given $x \in \{0, 1\}^l$, find a subset of the coalition that produced x. We denote by B_m is the set of all bit positions in which the column m is replicated, $R_m = B_{m-1} \cup B_m$ and weight denotes the number of bits that are set to 1.*
   (a) If weight $(x \mid B_1) > 0$ then output "User 1 is guilty"
   (b) If weight $(x \mid B_{n-1}) < d$ then output "User n is guilty"
   (c) For all $s = 2$ to $n - 1$ do:
   Let $k =$ weight $(x \mid R_s)$. if

$$weight(x \mid B_{s-1}) < \frac{k}{2} - \sqrt{\frac{k}{2} \log \frac{2n}{\epsilon}}$$

   then output "User s is guilty"

Finally, the only thing left to do is to find a relationship between the error ϵ and the replication factor d. This relation is given in the following theorem,

Theorem 1. *For $n \geq 3$ and $\epsilon > 0$ let $d = 2n^2 \log(2n/\epsilon)$. The fingerprinting scheme $\Gamma_0(n, d)$ is n-secure with ϵ-error and has length $d(n-1) = O(n^3 \log(n/\epsilon))$.*

3.2 Logarithmic Length *c*-Secure Codes

The construction of Boneh and Shaw n-secure with ϵ-error is impractical for a medium and large number of user because the length of codewords increases as $O(n^3 \log(n/\epsilon))$. To achieve shorter codes, Boneh and Shaw apply the ideas of [9] to construct c-secure (n, l)-codes of length $l = c^{O(1)} \log(n)$. The basic idea is to use the n-secure code as the alphabet which is used by an $(L, N, D)_p$-error-correcting code. As a result of this composition, Boneh and Shaw obtained the following result. The proof of this theorem can be found in [2].

Theorem 2. *Given integers N,c, and $\epsilon > 0$ set $n = 2c$, $L = 2c \log(2N/\epsilon$, and $d = 2n^2 \log(4nL/\epsilon)$. Then, $\Gamma'(L, N, n, d)$ is a code which is c-secure with ϵ-error. The code contains N words and has length $l = O(Ldn) = O(c^4 \log(N/\epsilon) \log(1/\epsilon))$*

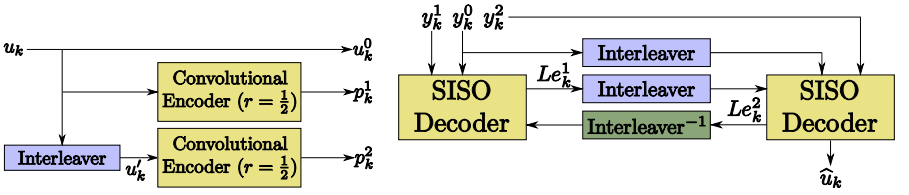
Thus the code $\Gamma'(L, N, n, d)$ is made up of L copies of $\Gamma_0(n, d)$. Each copy is called a *component* of $\Gamma'(L, N, n, d)$. The codewords of component codes will be kept hidden from the users. Finally, the codewords of $\Gamma'(L, N, n, d)$ are randomly permuted by π before the distributor embeds the codeword of the user u_i in an object, that is to say, user u_i's copy of the object will be fingerprinted using the word $\pi w^{(i)}$. To guarantee the security of this scheme, the permutation π must be kept hidden from the users in order to hide the information of which mark in the object encodes which bit in the code.

4 Turbo Codes

Turbo codes were introduced in 1993 by Berrou, Glavieux and Thitimajashima [10], [11]. In their research, they reported extremely impressive results for a code with a long frame length. The main idea is an extrapolation from Shannon's theory of communication. Shannon shows that an ultimate code would be one where a message is sent infinite times, each time shuffled randomly, but this requires infinite bandwidth so this schema is unpractical. The contribution of turbo codes is that sending the information infinite number of times is not really needed, just two or three times provides pretty good results.

4.1 Turbo Coding

The most common turbo encoder consists of parallel concatenation of some Recursive Systematic Convolutional encoders (RSC), each with a different interleaver, working on the same information. The purpose of the interleaver is to offer to each encoder an uncorrelated version of the information. This results in independent parity bits from each RSC. It seems logical that as a better interleaver is used, these parity bits will be more independent. The usual configuration consists of two identical convolutional encoders with rate 1/2 and a pseudo-random



(a) Common turbo encoder (b) Common turbo decoder

Fig. 1. Dual Turbo Encoder/Decoder with ratio $r = \frac{1}{3}$

interleaver, π, this schema is called a Parallel Concatenated Convolutional Code (PCCC). Figure 1(a) shows the block diagram of a turbo encoder with its two constituent convolutional encoders.

The input bits u are grouped in sequences whose length N is equal to the size of the interleaver. The sequence u' is obtained as the result of the interleaving process. The first encoder receives the sequence u and produces the pairs (u_k, p_k^1) and the second encoder receives the sequence u' and produces the pairs (u'_k, p_k^2). Since both encoders are systematic encoders $u'_k = \pi(u_k)$, and, as π is known by the decoder, only (u_k, p_k^1, p_k^2) will be transmitted. The rate of this encoder is 1/3 but it can be increased by puncturing by 1/2.

4.2 Turbo Decoding

Turbo decoding is based on an iterative process to improve performance and it uses, as a basic decoder unit, a Soft-Input Soft-Output algorithm. The block scheme of a common turbo decoder is shown in figure 1(b).

First of all, the sequence encoded by the first encoder is decoded by the first decoder as in an usual convolutional code scheme. As a result, this decoder returns soft information, that is to say, an estimation about which were the values of the bit in the original sequence and how likely is this estimation for each bit. This information is called extrinsic information in the literature of turbo codes. The extrinsic information of the first decoder is interleaved in the same manner that the input bits had been interleaved in the turbo encoder before they are applied to the second encoder. The next step is to send this interleaved information to the second decoder. This decoder takes the extrinsic information of the first decoder into account when it decodes the sequence encoded by the second encoder and gives a new estimation about the original values. This process is repeated several times depending on the performance that is required of the system. On the average, 7 or 8 iterations give adequate results and no more 20 are ever required.

There are some algorithms that can be modified to use as a turbo decoder component but the ones most used are the Soft Output Viterbi Algorithm [12,13] and the BCJR [14] or Maximum A-posteriori Probability (MAP) algorithm. SOVA is a combination of iterative decoding with a modified form of Viterbi decoding and it maximizes the probability of a sequence. On the other hand,

MAP maximizes the output probability based on some knowledge of the input *a priori* probabilities and soft output from the demodulator.

5 Turbo Fingerprinting Scheme

The major contributions of the turbo fingerprinting scheme, presented by Zhang *et al.* in [7] with regard to the Boneh-Shaw's scheme are the reduction of codeword length by means of the use of turbo codes as outer code and the improvement of decoding the decoding runtime by a Maximum Likelihood Decoding algorithm.

5.1 Concatenated Code

The proposed scheme consists of a concatenated turbo code with a Boneh-Shaw code, that is, each symbol that a turbo encoder generates is coded by a Boneh-Shaw encoder. Formally, Zhang *et al.* define their code $\Psi(L, N, n, d)$ as the concatenated code that results of the composition of an outer (n_0, k_0)-turbo code and an inner Boneh-Shaw $\Gamma_0(n, d)$-code, where L is a turbo code length and N is the users' number.

The first step in the process is to generate a random binary string that will be the user identification $m(u_i)$ for the user u_i, where $1 \leq i \leq N$. Next, $m(u_i)$ is divided into L groups of k_o bits each one. This groups are encoded by an (n_o, k_o)-turbo encoder and a sequence of $L \times n_0$ bits is produced. The output binary sequence is represented by $v = v_1 v_2 \cdots v_L$ where each group v_j is constituted by n_0 bits. Each v_j is coded by the inner code $\Gamma_0(n, d)$ where, for design reasons, n must satisfy the condition $n \geq 2^{n_0}$. As a result, the sequence $W^{(v)} = W^{(v_1)} \parallel W^{(v_2)} \parallel \cdots \parallel W^{(v_L)}$ is obtained, where $W^{(v_j)}$ is the codeword of $\Gamma_0(n, d)$-code assigned to v_j.

To formalize the encoding process, Zhang *et al.* define the $\Psi(L, N, n, d)$ encoding algorithm as follows:

Algorithm 2. *$\Psi(L, N, n, d)$ encoding algorithm defined in [7]:*
Let $m(u_j)$ be the identification of user u_j $(1 \leq j \leq N)$

1. $v = Turbo - Encoding(m(u_j))$
2. *For each* $1 \leq k \leq L$
   $W^{(v_k)} = \Gamma_0(n, d) - Encoding(v_k)$
3. *Let* $W^{(v)} = W^{(v_1)} \parallel W^{(v_2)} \parallel \cdots \parallel W^{(v_L)}$

This process is repeated for all u_j in such a way that all users will have their own identification fingerprint. In the fingerprinting environments the common attack is the collusion attack, that is, some users compare their marked objects and produce, according to the *Marking Assumption* defined in Definition 5, a pirate object which contains a false fingerprint that lies in $\Gamma(C)$. The general schema for two traitors is shown in Figure 2.

The aim of this kind of systems is to find, at least, one user who is part of the coalition. So the authors present Algorithm 3 to accomplish this purpose.

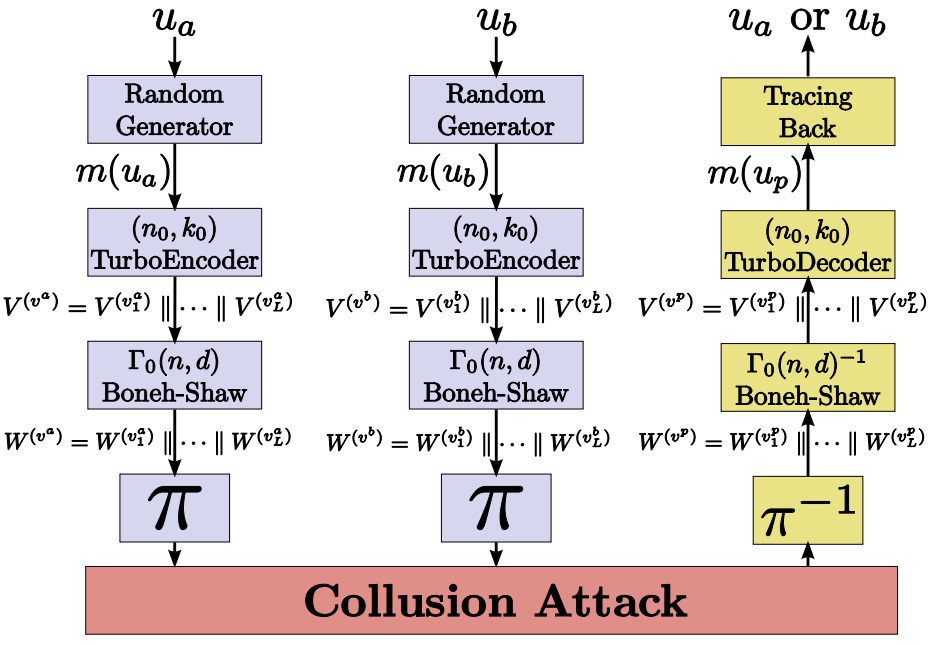


Fig. 2. Turbo fingerprinting scheme for 2 traitors

The main idea of the decoding algorithm is to decode the Boneh-Shaw layer and to choose one of the symbols retrieved by this layer for this position i as the input symbol to the turbo decoder for this position i. Note that, if the Boneh-Shaw code was error-free, then for each position, the turbo decoder could choose among more than one symbol, depending on the symbols of the traitors in this position. The proposal of Zhang *et al.* was to choose one at random. A formal definition is shown by the following algorithm:

Algorithm 3. *$\Psi(L, N, n, d)$ decoding algorithm defined in [7]:*
Given $x \in \{0,1\}^l$, find a subset of the coalition that produced x.

1. *Apply algorithm 2 to each of the L components of x.*
   For each component $i = 1, 2, \cdots, L$, arbitrarily choose one of the outputs of algorithm 2.
   Set v_j to be this chosen output.
   Form the word $v = v_1 v_2 \cdots v_L$
2. *$m(u_j) = Turbo - Decoding(v)$*
3. *Output "User u_i is guilty"*

6 A New Critical Performance Analysis

To state the performance of turbo fingerprinting codes, the authors in [7] enunciate the following theorem:

Theorem 3. *[7] Given integers N, c and $\epsilon > 0$, set*

$$n = 2c$$

$$d = 2n^2(\log(2n) + m)$$

$$m = \log\left(\sum_{d_e=d_{min}}^{N} \frac{A_{d_e}}{\epsilon}\right)$$

where A_{d_e} is the number of codewords with weight d_e. Then the fingerprinting scheme $\Psi(L, N, n, d)$ is c-secure with ϵ-error. The code contains N codewords and has length $Ld(n-1)$. Let x be a word which was produced by a coalition C of at most c users. Then algorithm 3 will output a member of C with probability at least $1 - \epsilon$.

The authors in [7] prove theorem 3, assuming the well known expression for the error probability P_e of turbo decoders in BSC channels (for detailed references concerning error probability of turbo decoders in BSC channels see [15,16]). In the present scenario the channel, from the turbo codes point of view, is a Boneh-Shaw code with error probability ϵ'. In [7], the authors express the turbo coded error probability as a function of the Boneh-Shaw code error probability. Denoting by P_e, the error probability of turbo codes in a BSC, the expression is

$$P_e \leq \sum_{d_e=d_{min}}^{N} A_{d_e} P_2(d_e) \tag{1}$$

where A_{d_e} is the number of codewords with weight d_e and $P_2(d_e)$ is the error probability between two codewords. Let the decoding error probability of code Γ_0 be ϵ'. The authors assume that the error probability between two codewords is smaller than the error probability of code Γ_0. So, from the authors' point of view

$$P_e \leq \sum_{d_e=d_{min}}^{N} A_{d_e} P_2(d_e) \leq \sum_{d_e=d_{min}}^{N} A_{d_e} \epsilon' \tag{2}$$

We now show that this is not in many cases correct.

Suppose a turbo fingerprinting code consists of an (n, k)-turbo code concatenated with a Boneh-Shaw code with negligible error probability ϵ. Moreover assume that two traitors attack this scheme by a collusion attack according to definition 5.

In the decoding process, the Boneh-Shaw decoder retrieves, for each position, 2 symbols with probability $\frac{2^n-1}{2^n}$ and only 1 symbol otherwise (here we suppose, as an approximation, that in a collusion of 2 users a position can not be detected with probability $\frac{1}{2^n}$. So $\frac{1}{2^n}$ is the probability that the symbol in a particular position will be the same for 2 codewords). The scheme proposed by Zhang *et al.* takes one of them at random and sends it to the turbo decoder.

As n increases, $\frac{2^n-1}{2^n}$ tends to 1, that is, the probability that the traitors, say tc_1 and tc_2, have the same symbol in a particular position tends to 0. So the

Hamming distances between tc_1 or tc_2 and the pirated word, say tc_p, delivered to the turbo decoder satisfy the equation $d_H(tc_1, tc_p) \simeq d_H(tc_2, tc_p)$. If L is the length of the codeword and n is large enough, the turbo decoder takes as input a word in which half of the symbols are erroneous respect both of the two traitor codewords. And, as a result, the turbo decoder retrieves a codeword of the turbo code codebook, but this codeword is not assigned to any user. Note that in this case, the decoded codeword will be different from the pirate words with a very high probability, which is not desirable at all. In this case there cannot be a false positives because, the turbo encoded words are a random sequence (like hash functions) and the collision probability for these functions is very small.

As an example suppose a $(3,1)$-turbo code that consists of two component convolutional codes. The connection expressed in octal is $(3,1)$. The traitors have the sequences

$$t_1 = 1\,1\,0\,0\,0\,1\,0\,0\,1\,0,$$

$$t_2 = 0\,0\,0\,0\,1\,1\,1\,0\,0\,0.$$

The sequences t_1 and t_2 are turbo coded to generate

$$tc_1 = Turbo - Encoding(t_1) = 100\,110\,000\,001\,001\,101\,011\,010\,110\,001\,000\,000,$$

$$tc_2 = Turbo - Encoding(t_2) = 000\,000\,000\,001\,100\,111\,101\,011\,011\,011\,111\,100.$$

These turbo coded sequences may be expressed in octal notation as:

$$tc_1 = Turbo - Encoding(t_1) = 4\,6\,0\,1\,1\,5\,3\,2\,6\,1\,0\,0$$

$$tc_2 = Turbo - Encoding(t_2) = 0\,0\,0\,1\,4\,7\,5\,3\,3\,3\,7\,4$$

The Feasible Set will be

$$\Gamma(tc_1, tc_2) = \left(\begin{Bmatrix}4\\0\end{Bmatrix}\begin{Bmatrix}6\\0\end{Bmatrix}, 0, 1, \begin{Bmatrix}1\\4\end{Bmatrix}, \begin{Bmatrix}5\\7\end{Bmatrix}, \begin{Bmatrix}3\\5\end{Bmatrix}, \begin{Bmatrix}2\\3\end{Bmatrix}, \begin{Bmatrix}6\\3\end{Bmatrix}, \begin{Bmatrix}1\\3\end{Bmatrix}, \begin{Bmatrix}0\\7\end{Bmatrix}, \begin{Bmatrix}0\\4\end{Bmatrix}\right).$$

After decoding the Boneh-Shaw code, if no errors are produced, a possible sequence sent to the turbo decoder is

$$tc_p = 000\,110\,000\,001\,001\,101\,101\,011\,011\,001\,111\,000,$$

or in octal,

$$tc_p = 0\,6\,0\,1\,1\,5\,5\,3\,3\,1\,7\,0.$$

After turbo decoding, the word obtained is

$$t_p = 1\,1\,0\,0\,0\,0\,1\,0\,0\,0,$$

which is none of the traitors' codewords, $d_H(t_1, t_p) = 3$ and $d_H(t_2, t_p) = 4$. That is, this construction cannot be a correct fingerprinting scheme because the system cannot trace back t_1 or t_2 from tc_p.

7 Proposed Improvements and Open Problems

One of the most important improvements that the turbo codes have contributed to error correcting codes is the use of the likelihood of every information bit during the decoding process. The proposed improvements in this section are based on the use of this information in two different ways. The first one, uses the information provided by the fingerprinting layer to calculate the likelihood for each information bit at the first turbo decoding iteration. On the other hand, the second proposal is centered in the fact that the cross-correlation between the likelihood of the decoded bits and all possible words (users) reaches the maximum value when the evaluated user has taken part in the collusion attack, i.e. is guilty.

7.1 Decoding by the Use of Likelihood Information in Undetected Coefficients

There exist some techniques, as concatenating a turbo codes with a Boneh-Shaw code or the ones proposed in [17], that can be used to detect, at the turbo decoder input, if a particular bit has been modified by a collusion attack. As it is also well-know, each constituent decoder in a turbo decoder uses the likelihood information of every information bit externalized by the other but this likelihood is not known at the first iteration by the first constituent decoder. The usual solution is to consider that all values are equally likely, that is to say, the value of $L_e^{(2)}$ for all bits in the first iteration is initialized to 0 (take into account that L_e is the Log-likelihood ratio). The first improvement is to modify the value of L_e taking into account the information of the Boneh-Shaw layer. The main idea is, as the undetected bits by the traitors during the collusion attack are known, the decoder can assign a greater likelihood to these bits in the first decoding stage. After few simulations, it can be concluded that a little improvement around 2% appears if the initial value of L_e is slightly modified by the use of this previous information. Note that, when an error is produced during the decoding process, the returned word identifies one legal user which has not taken part in the collusion, that is a false positive. In other words, in this situation the decoding process frames an innocent user. In a correct TFC system, a bit error probability around 0 is needed. If the value of L_e value is highly altered, the effect can be counterproductive because, in this case, the turbo decoder does not converge correctly. This is shown in figure 7.1.

Even though this improvement has been applied to TFC, it is not sufficient to guarantee that the probability of finding one of the traitors will be small enough. It seems that the error probability has been a slight improvement and it can be reduced near to 0 by the use of some block error correcting code as BCH. The figures 4(a), 4(b) and 4(c) show the results of the use of a (15,11) Hamming code, a BCH(127,64) and a BCH(127,22) respectively, concatenated with a turbo code decoded taking into account the likelihood information.

Some open problems are how to modify the channel characteristic, in turbo code notation that is the value of L_c, taking into account the positions not

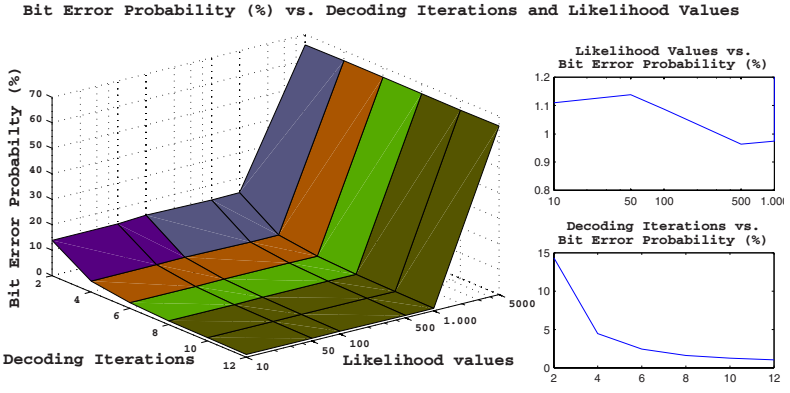


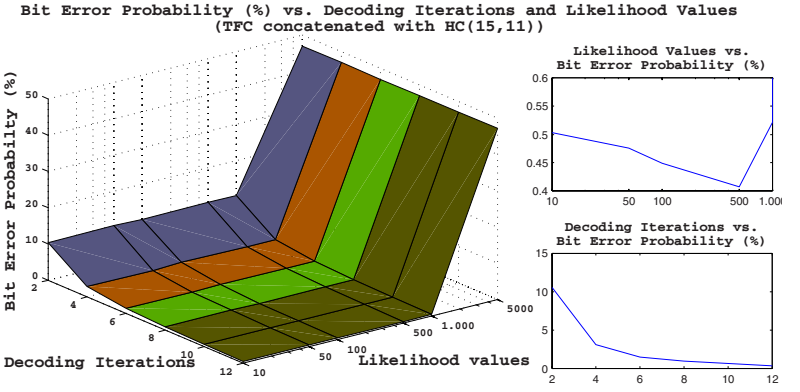
Fig. 3. Bit probability error of a TFC with generator sequences constituent RSC $(53, 75)_8$ over collusion attack decoded using likelihood information

detected by the attackers and the study of the relation between the RSC generator sequences and the likelihood value to assign to each bit at first decoding stage.

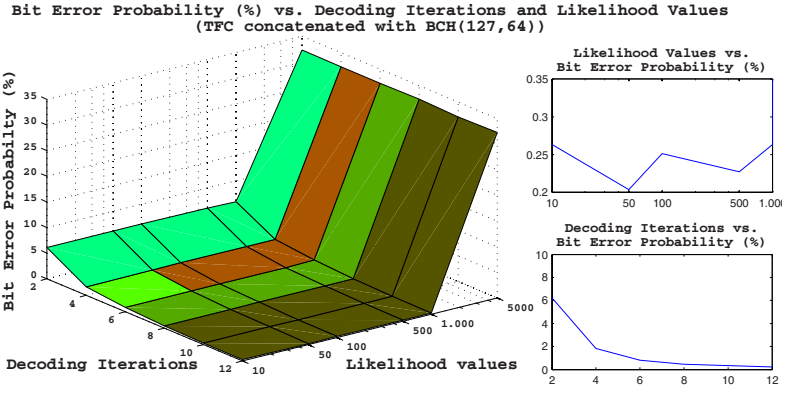
7.2 Decoding by the Use of Making Correlation Conditional on Likelihood

The decoding algorithm proposed in the original paper of TFC was the commonly used to decode turbo codes. The main problem was that the turbo decoder returns the most likely codeword over all the code space; that is, if a user ID of 512 bits is used, the turbo decoder will return the word of 512 bits that is the most likely to have been sent. This means that with a very high probability an innocent user has been framed. This is the main reason of the problems produced in the decoding stage of TFC.

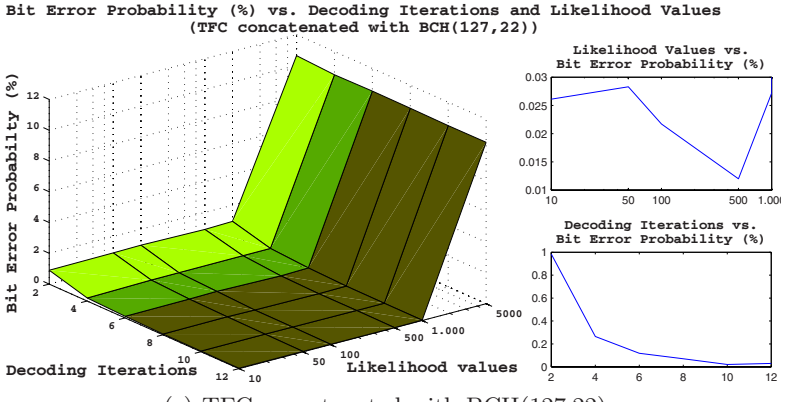
In practice, it is very difficult to think about an application in which 2^{512} users are needed. For instance, Figure 5 shows the results of a 1000 iterations simulation of one system which uses a TFC with user ID of 128 bits but the system has only 1000 users whose user IDs are randomly distributed in the codes pace. In each iteration two different users are chosen randomly and a collusion attack is performed with their code words. Next, this colluded codeword is decoded by the turbo decoder in order to obtain one of the traitors. None of them have been found by the use of the original TFC decoding system or, as is named in the figure, TFC without correlation. If the word which results from the TFC original decoding system is correlated with all possible user IDs, we will always find at least one of the traitors and, more than 90 percent of the times, the two traitors will be found. If the likelihood information is used instead of the pirate word, the probability of finding the two traitors comes close to 100 percent. In other words, the user IDs that have the maximum correlation value with the likelihood returned by the turbo decoder are the user IDs of the members of the collusion.



(a) TFC concatenated with HC(15,11).

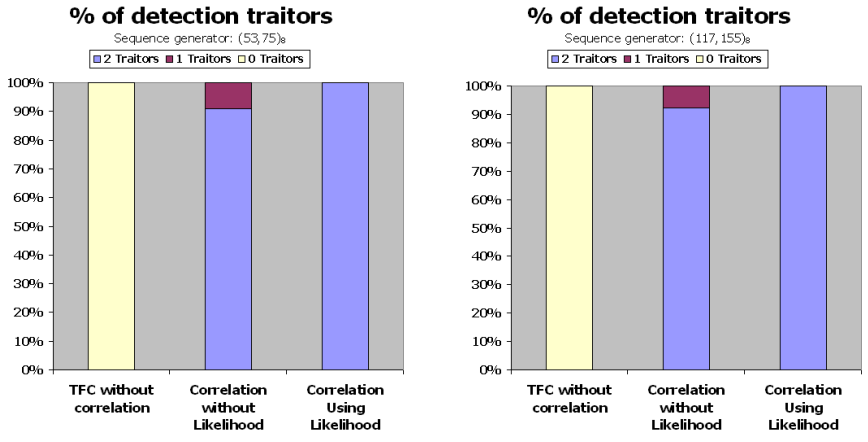


(b) TFC concatenated with BCH(127,64).



(c) TFC concatenated with BCH(127,22).

Fig. 4. Bit probability error of a TFC with generator sequences constituent RSC $(53, 75)_8$ concatenated with several error correcting codes over collusion attack decoded using likelihood information



(a) TFC with generator sequences constituent RSC $(53, 75)_8$

(b) TFC with generator sequences constituent RSC $(117, 155)_8$

Fig. 5. % of detecting 0, 1 or 2 traitors after a collusion attack of 2 traitors by the use of TFC with correlation decoding

The main drawback of this proposal is that the decoding time increases exponentially with the number of users.

8 Conclusions

The work presented in this paper discusses an undesired problem in the analysis of the turbo fingerprinting codes presented by Zhang *et al.* in [7]. We show that the probability of tracing one of the traitors tends to 0 when the alphabet size of the outer turbo code increases. That is because the symbol-by-symbol collusion attack performed by pirates is not treated efficiently by the decoding algorithm proposed in [7]. Note that, from the point of view of the turbo decoder, the error probability of the equivalent channel tends to 1/2, because it takes as input symbols one of the symbols retrieved by the Boneh-Shaw decoder chosen at random.

The new problem found in the turbo fingerprinting codes renders them inapplicable in many cases unless the design takes into account our new contribution. Moreover, our studies indicate that, the more efficient the turbo fingerprinting code design is, from the point of view of the length requirement, a far worse performance is obtained from the tracing algorithm. In other words, to find a traitor will be more complicated when the (n, k)-turbo code used, has large values of n.

Besides, two different ways to improve the performance of turbo fingerprinting codes are given. These two ways use the likelihood of the turbo decoder to perform the improvements. The first proposal modifies this likelihood at the input of the turbo decoder and the other use the turbo decoder output likelihood

to correlate it with the user IDs in order to find the traitors. Moreover, this two improvements can be integrated in the same scheme.

Acknowledgements

The authors wish to thank Juan Pérez-Esteban for helping in the simulations and the anonymous reviewers for their valuables comments.

References

1. Wagner, N.R.: Fingerprinting. In: SP 1983: Proceedings of the, IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 18. IEEE Computer Society, Los Alamitos (1983)
2. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data (extended abstract). In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 452–465. Springer, Heidelberg (1995)
3. Barg, A., Blakley, G.R., Kabatiansky, G.A.: Digital fingerprinting codes: problem statements, constructions, identification of traitors. IEEE Transactions on Information Theory 49(4), 852–865 (2003)
4. Fernandez, M., Soriano, M.: Fingerprinting concatenated codes with efficient identification. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 459–470. Springer, Heidelberg (2002)
5. Zhu, Y., Zou, W., Zhu, X.: Collusion secure convolutional fingerprinting information codes. In: ASIACCS 2006: Proceedings of the, ACM Symposium on Information, computer and communications security, pp. 266–274. ACM Press, New York (2006)
6. Tomàs-Buliart, J., Fernandez, M., Soriano, M.: Improvement of collusion secure convolutional fingerprinting information codes. In: Proceedings of International Conference on Information Theoretic Security (ICITS 2007) (2007)
7. Zhang, Z., Chen, X., Zhou, M.: A digital fingerprint coding based on turbo codes. In: International Conference on Computational Intelligence and Security, 2007, pp. 897–901 (2007)
8. Hamming, R.W.: Error detecting and error correcting codes. Bell System Techincal Journal 29, 147–160 (1950)
9. Chor, B., Fiat, A., Naor, M., Pinkas, B.: Tracing traitors. IEEE Transactions on Information Theory 46(3), 893–910 (2000)
10. Berrou, C., Glavieux, A.: Near optimum error correcting coding and decoding: turbo-codes. IEEE Transactions on Communications 44(10), 1261–1271 (1996)
11. Berrou, C., Glavieux, A., Thitimajshima, P.: Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In: IEEE International Conference on Communications, 1993. ICC 1993. Geneva. Technical Program, Conference Record, May 23-26, 1993, vol. 2, pp. 1064–1070 (1993)
12. Hagenauer, J., Hoeher, P.: A viterbi algorithm with soft-decision outputs and its applications. In: Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM 1989, November 27-30, 1989, vol. 3, pp. 1680–1686. IEEE, Los Alamitos (1989)
13. Hagenauer, J., Papke, L.: Decoding turbo-codes with the soft output viterbi algorithm (sova). In: Proceedings of the IEEE International Symposium on Information Theory (June 27–July 1, 1994), p. 164 (1994)

14. Bahl, L., Cocke, J., Jelinek, F., Raviv, J.: Optimal decoding of linear codes for minimizing symbol error rate (corresp.). IEEE Transactions on Information Theory 20(2), 284–287 (1974)
15. Lin, S., Costello, D.J.: Error Control Coding, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (2004)
16. Vucetic, B., Yuan, J.: Turbo codes: principles and applications. Kluwer Academic Publishers, Norwell (2000)
17. Navau, J.C., Fernandez, M., Soriano, M.: A family of collusion 2-secure codes. In: Barni, M., Herrera-Joancomartí, J., Katzenbeisser, S., Pérez-González, F. (eds.) IH 2005. LNCS, vol. 3727, pp. 387–397. Springer, Heidelberg (2005)