Capacity of collusion secure fingerprinting — a tradeoff between rate and efficiency

Gábor Tardos
School of Computing Science
Simon Fraser University
and
Rényi Institute, Budapest
tardos@cs.sfu.ca

Abstract

The talk presented a short history of collusion resistant fingerprinting concentrating on recent results that represent joint work with Ehsan Amiri.

1 The model

To ensure protection of their copyright, content producers often make each copy of their productions unique by embedding a distinct code in each. For this to work they have to be able to hide the positions where the code is embedded. A *collision attack* is performed by a group of malicious users (the *pirates*), who compare their copies and identify the positions where they differ as a position of the embedded code. They can then arbitrarily change the code in these positions. We assume however that they do not notice the positions of the hidden code where all their codes agreed and therefore they cannot alter these positions. This is the *marking assumption.*

A (collusion resistant) *fingerprinting code* consists of a randomized procedure to choose codewords (the *code generation*) and a *tracing algorithm* that finds one of the pirates based on all these codewords and and the *forged codeword* read from the unauthorized copy made by the pirates. We say that the code (or the tracing algorithm) *errs* if it falsely accuses an innocent user or outputs no accused user at all. This should happen with small probability. The mathematical definition was first given by Boneh and Shaw [4] and can also be found in most of the papers cited below. Here we do not go beyond the slightly informal explanation given above, but in order to be able to speak about the result we have to list the numerous parameters a fingerprinting code has. These are

- *alphabet size* The codewords are are sequences over a fixed alphabet Σ. Most of the research on fingerprinting codes concentrates on the binary alphabet $\Sigma = \{0, 1\}$, but fingerprinting is worth studying over larger alphabets too and the size $|\Sigma|$ of the alphabet is an important parameter.
- *codelength* This is the length of the codewords, usually denoted by n.
- *number of users* Usually denoted by N, this is also the number of codewords.
- *number of pirates* In most codes one has to assume a bound t on the number of pirates responsible for the collusion attack. If the actual number of pirates is t or less, the tracing algorithm should perform with small error probability. In many of the fingerprinting codes the probability of accusing an innocent user is small even if the number of pirates exceeds t, only the probability of tracing algorithm failing by not producing any accusations increases in this case.
- *error probability* We say that a a code is ϵ-secure against t pirates or it is an ϵ-secure t-fingerprinting code, if the probability of the error of the tracing algorithm is at most ϵ for any set of at most t pirates performing an arbitrary pirate strategy to produce the forged codeword provided that they obey the marking assumption.
- *rate* The rate R of a fingerprinting code is computed from the number N of codewords and the length n as $R = \log N/n$, where the logarithm is binary. In the trivial $t = 1$ case (no collusion) it is enough to ensure that the codewords are pairwise distinct and thus a rate of $R = \log |\Sigma|$ is achievable ($R = 1$ for binary codes), the reciprocal of the rate gives how much longer the codewords are compared to these trivial binary codewords.

To simplify the large number of parameters we concentrate on maximizing the rate of a sequence of fingerprinting codes (a *fingerprinting scheme*) subject to the conditions that the number of users and the length should go to infinity, the error probability should go to zero while the number of pirates and and the alphabet is fixed. The *t-fingerprinting capacity* for alphabet size $|\Sigma|$ is the maximum achievable limit rate of such fingerprinting schemes.

The goal of fingerprinting research is to find efficient and secure fingerprinting codes. The paramount problem in the application of fingerprinting codes is the high cost of embedding every single digit of the code. This makes it important to design secure fingerprinting codes that are short, or equivalently, have high rate. In particular, recent research focused on finding or estimating the t-fingerprinting capacity for various values of t (mostly considering the binary alphabet).

2 History of results

Boneh and Shaw [4] were first to define fingerprinting secure against collusion attacks. They proposed such binary codes of length $O(t^4 \log(N/\epsilon) \log(1/\epsilon))$ for N users that are ϵ-secure against t pirates. This translates to a t-secure fingerprinting scheme with rate of $\Omega(1/t^4)$. The same paper gave a lower bound of $\Omega(t \log(1/(t\epsilon)))$ for the length of the fingerprinting code with the same parameters. This does not quite translate to an upper bound on the binary t-fingerprinting capacity but still roughly correspond to an $O(1/t)$ bound.

The paper [11] introduced a new and more efficient codes. The rate of these binary t-secure codes are $1/(100t^2)$ and the same paper also gave a lower bound on the length of t-secure fingerprinting codes that roughly translates to an $O(1/t^2)$ upper bound on the rate for any alphabet size. This settled the order of magnitude for the t-fingerprinting capacity: it is $\Theta(1/t^2)$. The large constant factor between the lower and upper bound motivated further research and many subsequent papers (e.g., [3, 7, 8, 9, 10]) managed to improve the constant 100 in the construction with optimizing various parameters in the construction and/or better analysis in the estimate of the error probability.

Amiri and Tardos [1] and independently Huang and Moulin [5, 6] (for a broader class of models including the marking assumption model surveyed here) designed fundamentally different fingerprinting codes in an attempt to find the optimal rate, the fingerprinting capacity. While these new codes in the two papers are slightly different they are very similar and achieve identical rates. I conjecture that the rates achieved are optimal (i.e., they achieve the t-fingerprinting capacity) to the best of my knowledge this has not been fully proved yet.

3 Comparison of the techniques

Bias based code generation was introduced in [11]. This is a two phase process for generating the code words starting with picking iid. *biases* for each position from 1 to n and continuing with picking each digit of each codeword independently with the bias determined by the position of the digit. In the binary case a bias for position i is a real number $0 \leq p_i \leq 1$ and the digit i of a codeword x is picked with $P[x_i = 1] = p_i$. For larger alphabets the bias is an arbitrary distribution on Σ. The same code generation was used later in [1], but with a different distribution to choose the biases from.

The tracing algorithm in [11] is simple and efficient, whether a user is accused or not is determined by simple linear constraint, most notably it is determined by the codeword of the user, the forged codeword and the biases without regard to all the other codewords. This makes the tracing algorithm linear time in the size of the code matrix. In contrast, in the schemes of [1, 5] and also in the scheme of the earlier paper [2] of Anthapadmanabhan, Barg and Dumer the tracing algorithm has to consider each t-tuple of users to decide which user to accuse, thus the accusation of a user depends on the codewords of all other users too. This makes tracing rather inefficient.

4 Further research directions

An obvious research direction is to combine the efficiency of the tracing algorithm in [11] with the higher rates of [2] (for $t = 2$ pirates) and of [1, 6] (for any number of pirates). With Ehsan Amiri we achieved partial results in this direction. These results are yet to be published.

For the first non-trivial case when the code should be secure against only $t = 2$ pirates we can design a more efficient tracing algorithm for the fingerprinting code of [2, 1] (these have the same very simple code generation procedure for $t = 2$: each user receives an independent uniform random binary codeword). Although accusation of a user still depends on codewords of all other users too (this is unavoidable), the new tracing algorithm is linear time in the size of the codematrix. So in the case of two pirates the shorter codes and the faster tracing algorithms can be achieved simultaneously without having to compromise in either.

For more than two pirates our results are much more modest. First, we can achieve a small speedup of the tracing algorithm. Instead of considering all t-tuples of users for a tracing algorithm with N^t as the leading term in its running time we can slightly improve the exponent. This represent a slightly more efficient, but still optimal length fingerprinting codes. Another possible approach is to insist on a much faster tracing algorithm for the price of a slightly longer code (i.e., lower rates). We devised a sequence of intermediate fingerprinting codes representing a gradual trade-off between efficiency (speed of tracing) and length (rate). It is not clear however, whether this trade-off is inherent or just the result of our imperfect techniques.

Another important research direction is to study how the alphabet size influences the fingerprinting capacity. By the results of [11, 1] the $O(1/t^2)$ upper bound for the t-fingerprinting capacity holds with an absolute constant independent of the alphabet size, but our preliminary calculations show that the actual t-fingerprinting capacity grows substantially with moving from binary to larger alphabets. It would be important to determine the limit rates achievable as the alphabet size grows.

References

[1] E. Amiri, G. Tardos, High rate fingerprinting codes and the fingerprinting capacity, in *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009)*, 336–345.

[2] N.P. Anthapadmanabhan, A. Barg, I. Dumer, Fingerprinting capacity under the marking assumption, *IEEE Transactions on Information Theory* **54** (6) (2008), 2678–2689. Preliminary version appeared in the Proceedings of the 2007 IEEE International Symposium on Information Theory, (ISIT 2007), 2007.

[3] O. Blayer, T. Tassa, Improved versions of Tardos' fingerprinting scheme, *Designs, Codes and Cryptography* **48** (2008), 79–103.

[4] D. Boneh, J. Shaw, Collusion-secure fingerprinting for digital data, *IEEE Transactions of Information Theory* **44** (1988), 480–491.

[5] Y. Huang, P. Moulin, Saddle-point solution of the fingerprinting capacity game under the marking assumption, in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009).*

[6] P. Moulin, Universal fingerprinting: capacity and random-coding exponents, in *Proceedings of the IEEE International Symposium on Information Theory (ISIT 2008)*, 220–224.

[7] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa, H. Imai, An improvement of the discrete Tardos fingerprinting codes, *Cryptology ePrint Archive*, Report 2008/338.

[8] K. Nuida, M. Hagiwara, H Watanabe, H. Imai, Optimization of Tardos's fingerprinting codes in a viewpoint of memory amount, in: *Information Hiding*, LNCS 4567, Springer Berlin, Heidelberg, 2008, pp. 279–293.

[9] B. Škorić, S. Katzenbeisser, M.U. Celik, Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes, *Designs, Codes and Cryptography* **46**(2) (2008), 137–166.

[10] B. Škorić, T.U. Vladimirova, M. Celik, J.C. Talstra, Tardos fingerprinting is better than we thought, *IEEE Transactions on Information Theory* **54** (8) (2008), 3663–3676.

[11] G. Tardos, Optimal probabilistic fingerprint codes, *Journal of the ACM*, to appear. Preliminary version appeared in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, (STOC 2003), 116–125.