Dynamic Traitor Tracing Schemes, Revisited

Thijs Laarhoven [#1]

[#] *Department of Mathematics and Computer Science*
Eindhoven University of Technology, The Netherlands
[1] mail@thijs.com

***Abstract*—We revisit recent results from the area of collusion-resistant traitor tracing, and show how they can be combined and improved to obtain more efficient dynamic traitor tracing schemes. In particular, we show how the dynamic Tardos scheme of Laarhoven et al. can be combined with the optimized score functions of Oosterwijk et al. to trace coalitions much faster. If the attack strategy is known, in many cases the order of the code length goes down from quadratic to linear in the number of colluders, while if the attack is not known, we show how the interleaving defense may be used to catch all colluders about twice as fast as in the dynamic Tardos scheme. Some of these results also apply to the static traitor tracing setting where the attack strategy is known in advance, and to group testing.**

I. Introduction

Suppose a distributor of streaming digital content wishes to sell his content to customers. To do this, he may encrypt the stream and grant paying customers access to the live stream to prevent others from freely accessing the data. However, a malicious customer may try to broadcast the stream directly to non-paying users by creating a pirate broadcast, possibly making money out of it as well. To prevent this, the distributor may embed unique watermarks into the content for each user, such that when a pirate streams his copy of the content online, the distributor may detect it, extract the watermark from the content, and see who is the pirate.

Things become more complicated when several pirates *collude* and mix their copies of the content into a new stream, with a sequence of watermarks that is not equal to any of their individual sequences of watermarks. Under the assumption that in each segment, these colluders can only output one of their watermarked segments (known as the marking assumption), it is still possible to trace collusions of any size c, by carefully choosing which watermarked segments each user should get, and designing an algorithm that traces (the watermark of) the pirate stream to the set of colluders. In the setting of live streams, the assignment of watermarks for the next content segments may be adjusted based on the previous pirate output; we call this the *dynamic* traitor tracing setting. The scenario where the assignment of watermarks to users has to be done completely in advance, and where users may only be disconnected from the stream at the end, is often referred to as *static* traitor tracing. The latter scenario may for instance apply to video on demand.

WIFS'2013, November 18-21, 2013, Guangzhou, China.
ISBN 978-1-4673-5593-3 *©2013 IEEE.*

The goal of collusion-resistant traitor tracing is to use as few watermarked content segments as possible, while still being able to trace the users guilty of creating the pirate broadcast.

A. Model

The above dynamic traitor tracing problem is often translated to the following mathematical model. First, we assume there is a set of n users, c of which are colluding to create the pirate output. We assume that in each content segment, the distributor is able to embed up to q different watermarks, with $q \ll n$ often referred to as the alphabet size. To each segment and each user, the distributor assigns one of q watermarks. This assignment is denoted by a code matrix $X = (X_{j,i})$, where $X_{j,i} = k$ means that in segment i, user j receives the kth (out of q) watermark.

After distributing the ith watermarked segments to all users, the set of colluders chooses one of their segments and outputs it, along with its watermark. We denote the sequence of watermarks in the pirate output by a vector $y = (y_i)$. Then, the distributor detects y_i, and uses this information to possibly disconnect some users that he thinks are guilty. After this, he chooses the next content segments $X_{j,i+1}$, possibly based on the previous pirate output y_i. This continues until no pirates are left to output one of their segments, or until we have reached the limit of the number of watermarked segments. We say a scheme is successful if, within an a priori fixed amount of segments ℓ, it is able to guarantee (with high probability) that all colluders are caught, and no innocent users are accidentally disconnected along the way. We write ε_2 for the probability of not catching all colluders, and ε_1 for the probability of ever disconnecting at least one innocent user.

B. Related work

Fiat and Tassa [1] showed that if the alphabet size q satisfies $q \geq 2c + 1$, then with $\ell = O(c \log n)$ segments, one can find and disconnect all colluders deterministically, i.e., with probability of error $\varepsilon_1 = \varepsilon_2 = 0$. This scheme is very efficient, and the only drawback is that it requires a large alphabet size, which may not be possible in practice. Tassa [2] later showed that combining the binary ($q = 2$) static scheme of Boneh and Shaw [3] with the dynamic scheme of Fiat and Tassa [1] leads to a binary dynamic traitor tracing scheme that uses $\ell = O(c^4 \log n)$ content segments. Due to the large required length of the code, this scheme is not practical.

Recently, Laarhoven et al. [4] showed that the celebrated binary static scheme of Tardos [5] can be converted into

a dynamic binary scheme, with a code length of the order $\ell = O(c^2 \log n)$. Together with the divide-and-conquer construction of Laarhoven et al. [6], allowing for a linear trade-off between the alphabet size and the code length, this allows one to build q-ary schemes with a code length of the order $\ell = O(\frac{c^2}{q} \log n)$, for arbitrary values of q. For large $q \approx O(c)$, this approaches the result of Fiat and Tassa, both in the alphabet size and in the number of segments needed.

Even more recently, Oosterwijk et al. [7] studied the score function used in Tardos' static scheme, and showed how to choose better score functions when the pirate attack is known. In one particular case, they came across a score function that turned out to work well against *any* attack. More precisely, for asymptotically large c, they showed that this score function achieves capacity, i.e., attains the known exact lower bound on the code length of $\ell \sim 2c^2 \ln n$ in the binary setting [8]. So for large c, using this new score function in the static setting may lead to a decrease in the length of the code of almost 60% compared to using the symmetric score function of Škorić et al. [9], for which it is known that the asymptotic code length $\ell \sim \frac{1}{2}\pi^2 c^2 \ln n \approx 4.93 c^2 \ln n$ is optimal [10].

C. Contributions and outline

The outline of this paper is as follows. First, in Section II we will revisit the dynamic Tardos scheme, and show how slightly modifying the scheme leads to a simpler proof and a slightly improved performance. Then, in Section III we take a closer look at the recent results of Oosterwijk et al. and show when and how these results can be improved and combined with the construction of the dynamic Tardos scheme to obtain efficient dynamic traitor tracing schemes. More precisely: we get a greatly improved performance when the attack strategy is completely known, as the order of the code length often goes down from quadratic in c to linear in c; and we heuristically show how to get the above-mentioned decrease in the asymptotic code length of up to 60% in dynamic traitor tracing, when no assumptions are made on the pirate attack. Finally, in Section IV, we summarize the results and mention some directions for future work.

In this paper we restrict our attention to binary, dynamic schemes, although some of the results may also be generalized to arbitrary alphabets, or applied to static traitor tracing.

II. The Dynamic Tardos Scheme, Revisited

Let us first recall the dynamic Tardos traitor tracing scheme of Laarhoven et al. [4], outlined below. The main difference between the celebrated static Tardos scheme [5] and the dynamic version of this scheme, is that while in the static scheme we disconnect a user only when his final score exceeds a certain threshold Z, in the dynamic scheme we disconnect users as soon as their scores exceed Z. With this small adjustment, it can be shown that whereas in the static Tardos scheme *at least one* pirate is caught with high probability, in the dynamic scheme *all* pirates are caught with high probability.

The construction below depends on scheme parameters ℓ, Z, and δ (a cutoff parameter on the distribution function F_δ). Choosing these parameters appropriately depends on whether one wants provable security or heuristic security. The provable bounds on ε_1 and ε_2 are not very tight, as the actual error probabilities are often significantly smaller. Heuristically, one may therefore be able to use shorter code lengths ℓ, at the risk of not satisfying provable upper bounds. In this paper we will not go into detail about choosing ℓ, Z, and δ; see e.g. [4], [11], [12], [13] for theoretical upper bounds, and [14], [15] for practical estimates of ℓ.

1) **Codeword generation**
   For each position $1 \le i \le \ell$:
   a) Select $p_i \in [\delta, 1-\delta]$ from $F_\delta(p)$ defined as:
   $$F_\delta(p) = \frac{2\arcsin(\sqrt{p}) - 2\arcsin(\sqrt{\delta})}{\pi - 4\arcsin(\sqrt{\delta})}. \qquad (1)$$
   b) Generate $X_{j,i} \in \{0,1\}$ using $P(X_{j,i} = 1) = p_i$.
2) **Distribution/Detection/Accusation**
   For each position $1 \le i \le \ell$:
   a) Send to each active user j symbol $X_{j,i}$.
   b) Detect the output y_i; terminate if there is none.
   c) Calculate scores $S_{j,i} = h(X_{j,i}, y_i, p_i)$ using:
   $$h(x,y,p) = \begin{cases} +\sqrt{p/(1-p)}, & (x,y) = (0,0), \\ -\sqrt{p/(1-p)}, & (x,y) = (0,1), \\ -\sqrt{(1-p)/p}, & (x,y) = (1,0), \\ +\sqrt{(1-p)/p}, & (x,y) = (1,1). \end{cases} \qquad (2)$$
   d) For active users j, set $S_j(i) = S_j(i-1) + S_{j,i}$, and disconnect this user if $S_j(i) > Z$.

By making the following small modification, one can get a slight improvement in the theoretical bounds on ℓ of [4] and get a simplified proof. The main difference is that scores are only updated in segments where no one is disconnected.

2d) If there are active users with $S_j(i-1) + S_{j,i} > Z$:
- Disconnect these users.
- For other active users, set $S_j(i) = S_j(i-1)$.

Else set $S_j(i) = S_j(i-1) + S_{j,i}$ for all active j.

Due to lack of space we omit the details, but with this modification, the proof will not have to resort to using the technical parameter $\tilde{Z}$ which is used in the proofs in [4]. It now suffices to prove that in positions where scores were calculated, the average pirate score will exceed Z. Both error probabilities $\varepsilon_1, \varepsilon_2$ still increase by (at most) a factor 2, and the code length increases by an additive term c, but otherwise we can use the same parameters as in the static Tardos scheme. This modification thus leads to a cleaner reduction between code lengths in the static ($\ell^{(s)}$) and dynamic ($\ell^{(d)}$) Tardos schemes as follows, with notation similar to [6]:

$$\ell^{(d)}(c, n, \varepsilon_1, \varepsilon_2) = \ell^{(s)}(c, n, \tfrac{1}{2}\varepsilon_1, \tfrac{1}{2}\varepsilon_2) + c. \qquad (3)$$

To give an indication of the provable code lengths with and without this modification: in the example of [4] with scheme

parameters $c = 25$, $n = 10^6$, $\varepsilon_{1,2} = 10^{-3}$, the number of segments now increases from 109 585 (static) to only 114 204 (dynamic), instead of the 116 561 of [4].

III. Optimal Score Functions, Revisited

Recently, Oosterwijk et al. [7] showed how to obtain optimized score functions for any attack strategy, maximizing the normalized expected coalition score $\tilde{\mu}$ per segment[1]. For several attacks, they showed that this leads to much more efficient schemes, sometimes reducing the order of the code length from c^2 to as little as $c^{1.5}$ when using the arcsine distribution for generating biases p_i. As a side result, they noticed that the score function built against the interleaving attack (randomly choosing a pirate, and outputting his watermarked segment) performs well against any attack.

In this section, we will discuss these improved score functions, as well as the optimal distribution functions against these attacks, in several different settings. We first discuss the scenario where the complete attack strategy is either known to the distributor in advance or can be accurately determined along the way [16], and how drastically the code lengths may be reduced in this setting. Then we will discuss the scenario where the attack strategy is not known and may not even be the same in each segment, in which case we show how we may be able to use the interleaving defense to our advantage.

A. Known attacks

If the attack strategy is completely known, we can not only use the optimized score functions of Oosterwijk et al. [7] to increase the performance, but also fix the value of p in advance to its optimal value. After all, if the attack strategy is known and identical for every column, we can compute the performance of the scheme as a function of p, and choose the value of p that maximizes it. Note that using different values of p for different columns is required in the ordinary traitor tracing setting, as an adversary could otherwise exploit this knowledge about p to build strong attacks. But if we already know what the attack looks like in advance, then the model should be considered a randomized model rather than an adversarial model, and varying p is no longer necessary.

Table I lists some common attacks also discussed in [7] and the optimal choices of h and p for each of these attacks, together with the asymptotic scaling of the lengths of the codes in each of these cases. Except for against the interleaving attack, in each of these examples the order of the number of segments needed to trace collusions goes down from $O(c^2 \log n)$ to $O(c \log n)$, thus leading to a drastic reduction in the code length. Below we briefly discuss each of these attacks, and give some intuition behind the results in the table. Due to space limitations, details on the optimal values of p, h, and the asymptotics of ℓ are omitted here but will appear in the full version of [17].

[1]Here, normalized refers to dividing by the standard deviation of innocent user scores per segment, which we will denote by σ.

1) The interleaving attack: Against the interleaving attack, the leading term of the code length does not depend on the choice of p when the interleaving defense is used [7, Proposition 10], and its asymptotic value is always given by $\ell \sim 2c^2 \ln n$. However, minimizing the variance of the coalition score by choosing $p = \frac{1}{2}$ leads to slightly smaller first order terms. In that case, the score function reduces to $+1$ for matches and -1 for differences.

2) The all-1 attack: For the all-1 attack (output a 1 if at least one of the colluders received a 1), the optimal choice of p is given by $p = \frac{\alpha}{c}$, where $\alpha = o(1)$ should not converge to 0 too quickly, to prevent first order terms from blowing up [17]. The all-1 attack is directly related to group testing [17], [18] where testing a pool of samples returns a positive result iff at least one positive item is present in the tested pool. Note that we have manually adjusted the function of [7] from $h(1,0) = -1$ to $h(1,0) = -\infty$, since if a user has a 1 and the pirate output is a 0, he is certainly innocent. The asymptotic code length is given by $\ell \sim 2c \ln n$ for this choice of p, while the simpler choice $p = \frac{1}{c}$ leads to $\ell \sim \frac{2e(e-1)}{2e-1} c \ln n$.

Note that the all-0 attack is equivalent to the all-1 attack with the interpretation of the symbols 0 and 1 reversed. So we can obtain the optimal h and p by replacing each occurrence of 0 by 1 and p by $1 - p$, and vice versa.

3) The coin-flip attack: Similar to the all-1 attack, the optimal choice of p against the coin-flip attack (randomly select either a 0 or a 1) is of the order $p = o(\frac{1}{c})$, leading to an asymptotic number of segments of $\ell \sim 4c \ln n$ for large n. Again, smaller values of p lead to larger first order terms, and so a compromise could be to take $p = \frac{1}{c}$ with an asymptotic code length of $\ell \sim 2(e^2 - 1)c \ln n$.

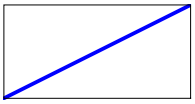
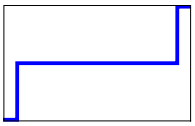
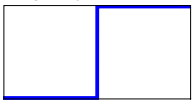
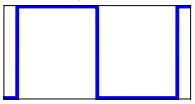
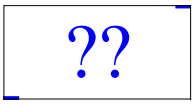
4) The majority voting attack: For arbitrary values of p and c, the optimal score function tailored against the majority voting attack (outputting the most common version among the pirate copies) looks quite nasty. However, optimizing p leads to $p = \frac{1}{2}$, leading to the trivial but intuitive score function of $+1$ for matches and -1 for differences. Determining $\tilde{\mu}$ turns out to be related to the expected absolute distance from the origin after c steps in a simple 1-dimensional random walk, and working out the details leads to an asymptotic code length of $\ell \sim \pi c \ln n$.

5) The minority voting attack: Again, for arbitrary values of p and c, the optimal score function built against the minority voting attack (always outputting the least common version among the pirate copies) does not look very pretty. However, the optimal choice of p is close to 0, in which case the minority voting attack is almost equivalent to the all-1 attack. The score function in the Table (which is the all-1 score function without manually optimizing h) is not exactly correct, but for reasonably large values of c and small values of p, the difference is negligible. For $p = o(\frac{1}{c})$ the optimal scaling of ℓ is equivalent to that of the all-1 attack, while for $p = \frac{1}{c}$ it is slightly higher due to the manual optimization of the score function of the all-1 attack.

Other attacks: Finally, for arbitrary other attacks not mentioned above, one can manually compute the optimal param-

TABLE I

OPTIMAL PARAMETER CHOICES FOR SEVERAL ATTACK STRATEGIES. THE PLOTS ON THE LEFT SHOW $P(y=1 \mid k)$ AGAINST k, WHERE $k \in \{0,\dots,c\}$ IS THE NUMBER OF ONES RECEIVED BY THE COALITION, WHEN THE CORRESPONDING ATTACKS ARE USED BY THE COALITION.

Attack	**Optimal score function** h		**Optimal bias** p	**Asymptotics** ℓ
Interleaving attack 	$h(x,y) = \begin{cases} +p/(1-p) \\ -1 \\ -1 \\ +(1-p)/p \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p=\frac{1}{2}$ $\left(p \in (0,1),\right.$	$\ell \sim 2c^2 \ln n$ $\left.\ell \sim 2c^2 \ln n\right)$
All-1 attack 	$h(x,y) = \begin{cases} +p/(1-p) \\ -p(1-p)^{c-1}/(1-(1-p)^c) \\ -\infty \\ +(1-p)^c/(1-(1-p)^c) \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p=o\left(\frac{1}{c}\right)$ $\left(p=\frac{1}{c},\right.$	$\ell \sim 2c \ln n$ $\left.\ell \sim \frac{2e(e-1)}{2e-1} c \ln n\right)$
Coin-flip attack 	$h(x,y) = \left(p^{c-1}+(1-p)^{c-1}\right) \times \begin{cases} +p/(1-p^c+(1-p)^c) \\ -p/(1+p^c-(1-p)^c) \\ -(1-p)/(1-p^c+(1-p)^c) \\ +(1-p)/(1+p^c-(1-p)^c) \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p=o\left(\frac{1}{c}\right)$ $\left(p=\frac{1}{c},\right.$	$\ell \sim 4c \ln n$ $\left.\ell \sim 2(e^2-1)c \ln n\right)$
Majority attack 	$h(x,y) = \begin{cases} +1 \\ -1 \\ -1 \\ +1 \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p=\frac{1}{2}$	$\ell \sim \pi c \ln n$
Minority attack 	$h(x,y) \approx \begin{cases} +p/(1-p) \\ -p(1-p)^{c-1}/(1-(1-p)^c) \\ -1 \\ +(1-p)^c/(1-(1-p)^c) \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p=o\left(\frac{1}{c}\right)$ $\left(p=\frac{1}{c},\right.$	$\ell \sim 2c \ln n$ $\left.\ell \sim 2(e-1)c \ln n\right)$
Unknown attacks 	$h(x,y,p) = \begin{cases} +p/(1-p) \\ -1 \\ -1 \\ +(1-p)/p \end{cases}$	$(x,y)=(0,0)$ $(x,y)=(0,1)$ $(x,y)=(1,0)$ $(x,y)=(1,1)$	$p \sim F_\delta$	$\ell \lesssim 2c^2 \ln n$

eters as follows:

- Compute the optimal score function h [7, Cor. 7 and 8].
- Determine the performance indicator $(\sigma^2/\tilde{\mu}^2)$ as a function of p, and choose the value of p that minimizes it.

Choosing ℓ, Z, and δ as a function of h, p, and c is not so easy, and as mentioned before depends on whether one wants provable security or only heuristic bounds on the error probabilities. Doing this analysis is left as an open problem.

Remark 1: Looking at the examples above, we notice that the optimal choice of p seems to be closely related to where the pirate strategy has the biggest jump in the probability of outputting a 1 (cf. the graphs in Table I). For instance, in the all-1 attack, this jump occurs between $\frac{k}{c}=0$ and $\frac{k}{c}=\frac{1}{c}$, and the optimal choice of p is indeed between 0 and $\frac{1}{c}$. When the pirates use the majority voting attack, a jump only occurs when the fraction of ones goes from slightly below $\frac{1}{2}$ to slightly above $\frac{1}{2}$, and indeed the optimal choice of p is exactly $\frac{1}{2}$. Note that except for against the majority attack, the asymptotic code length scales as $\ell \sim 2c\ln(n)/\beta$, where β is the size of the biggest jump. For instance, for the all-1 attack and minority attack, we have $\beta=1$; for the coin-flip attack we have $\beta=\frac{1}{2}$; and for the interleaving attack we have $\beta=\frac{1}{c}$. This also illustrates why the interleaving attack is very strong.

Remark 2: The above results can also be applied to the static traitor tracing setting to reduce the number of segments needed to catch colluders, but also to guarantee that *all* colluders are caught with high probability. In fact, the above asymptotics for ℓ also hold for static traitor tracing if $c=n^{o(1)}$, although the convergence to the limit is considerably slower. For instance, the above parameters for the all-1 attack lead to improved non-adaptive (static) group testing schemes. For details, see [17].

B. Unknown attacks

If the attack strategy is *not* known to the distributor, he has to choose h and the distribution function F such that *any* attack can be dealt with efficiently. An obvious suggestion would be to use the dynamic Tardos scheme with the score function replaced by the interleaving defense, as the interleaving defense

is asymptotically optimal. However, we run into some practical issues when we try to do this, since the interleaving defense does not satisfy the same nice properties as the Tardos defense. If we use the Tardos score function, we know that $\sigma^2 = 1$, and so we can fix the threshold Z in advance (based only on c, n, ε_1, ε_2) and get the proof methods of [4] to work. But if we use the interleaving defense, we only know that $\sigma^2/\tilde{\mu}^2 \leq 1$ for arbitrary attacks and large c, but we do not have any accurate estimates on σ^2. In fact, this parameter not only depends on the scheme parameters, but also on the attack strategy, which is unknown. For instance, using the minority attack leads to large values of both σ and $\tilde{\mu}$, while the majority attack leads to small values for both. Thus, using a fixed threshold Z does not seem to work.

1) A heuristic construction: We present a heuristic solution as follows, where the trick is that we vary Z throughout, depending on σ. During the scheme, we keep track of the average innocent standard deviation per segment $\bar{\sigma}(i) = \sqrt{\frac{1}{i}\sum_{k\leq i}\sigma_k^2}$, where $\sigma_k^2 = p_k h(1, y_k, p_k)^2 + (1-p_k)h(0, y_k, p_k)^2$ can be computed based on p_k and y_k. Then, after each position i, instead of using Z as a threshold, we use $Z(i) = Z \cdot \bar{\sigma}(i)$ as the threshold for disconnecting active users. The rest of the scheme remains the same.

In Figure 1 we show some experiments using this construction, where we used the same parameters ℓ, Z, and δ as in the dynamic Tardos scheme (which could be considered a conservative choice). Several attacks are considered with extreme values and fluctuations of σ, but in each case the scheme seems to work well, as all pirates are caught sooner than in the original dynamic Tardos scheme.

2) Possible issues: Although the scheme seems to work well, we mention some theoretical issues with this scheme.

a) The Gaussian assumption: Whereas in the Tardos scheme scores in each segment are identically distributed, the same does not hold for the interleaving defense. Some segments inevitably contribute much more to a user's overall score than other segments, due to varying values of p. Therefore, the assumption that the total user scores behave somewhat Gaussian may not hold, and the tails of the score curves may be bigger than expected. For instance, in Figures 1d, 1e, and 1f, some positions cause both the green and the red curves to go up more that in other segments. Note that this issue exists for the interleaving defense in general, i.e., also for the interleaving defense applied to static traitor tracing.

b) Asymmetric attacks: Related to the above, colluders may use column-asymmetric attacks (i.e., change their strategy over time) that cause large fluctuations in the parameters $\tilde{\mu}$ and σ over time. An attack may for instance consist of first using the minority attack to cause σ to go up, and then using the majority attack to make the value of Z go down. This could possibly lead to innocent users being disconnected, because of their initially high scores. Figures 1e, 1f suggest that such attacks are not very effective, but again, simulations do not guarantee that there is no other way to exploit this issue.

In both cases, choosing the cutoff δ sufficiently large seems crucial. In the original Tardos scheme things may already go wrong if the cutoff is too small, but in the case of the interleaving defense, the effects of small values of p are much more extreme. Not using a cutoff may lead to explosions of user scores when p happens to be very small and the pirates still manage to output a 1, and in general a too small cutoff may cause huge fluctuations in σ and $\tilde{\mu}$ in different segments. But if we do use an appropriate cutoff, as we did in Figure 1, these issues might not cause any practical problems at all.

Finally, note that if we were to normalize the scores in each segment, to avoid the two issues mentioned above, we end up with the Tardos score function again. So using the Tardos score function may still be a practical choice to guarantee theoretical security and avoid these possible issues mentioned above. But heuristically, using the interleaving defense seems pretty safe, and can lead to drastic reductions in the code length.

IV. Conclusion

To summarize the most important contributions of this paper:

- By making a small modification to the construction of the dynamic Tardos scheme, we get a simplified proof and a slightly improved performance.
- Against known attacks, optimizing p and h may lead to much faster tracing of collusions. The asymptotic code length often decreases from quadratic to linear in c.
- Against unknown attacks, we can use the interleaving defense and the arcsine distribution with cutoffs to catch arbitrary collusions much faster, by varying Z throughout.
- Some theoretical issues exist with the interleaving defense, but experimentally these issues seem less relevant. However, using sufficiently large cutoffs seems crucial.

An open problem remains whether the interleaving defense can provably withstand arbitrary attacks. For now, it seems that using the interleaving defense is quite safe and very efficient.

Acknowledgment

The author is grateful to Jan-Jaap Oosterwijk and Benne de Weger for their valuable comments and suggestions.

References

[1] A. Fiat and T. Tassa, "Dynamic traitor tracing," *Journal of Cryptology*, vol. 14, no. 3, pp. 354–371, 2001.

[2] T. Tassa, "Low bandwidth dynamic traitor tracing schemes," *Journal of Cryptology*, vol. 18, no. 2, pp. 167–183, 2005.

[3] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1897–1905, 1998.

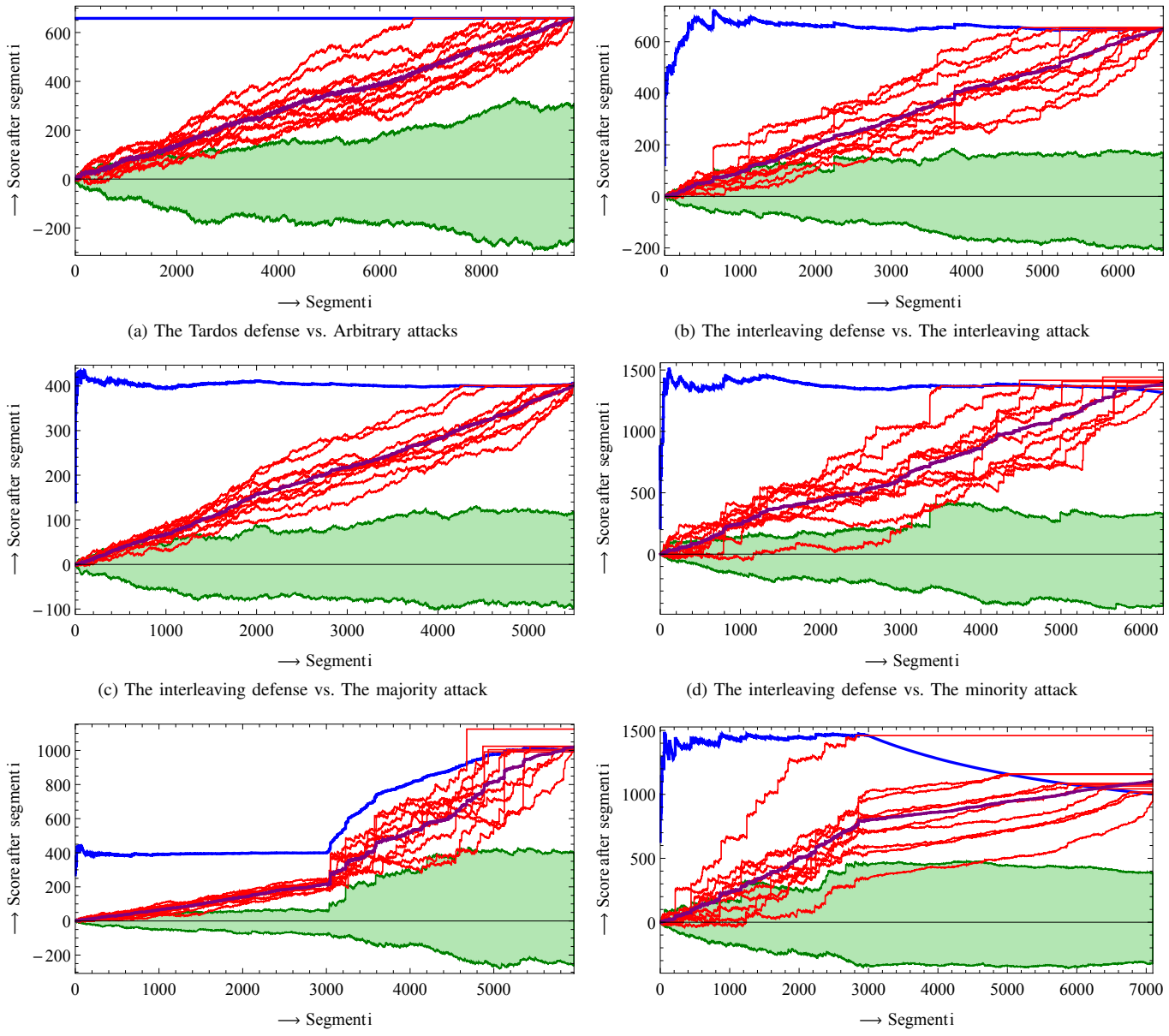
[4] T. Laarhoven, J. Doumen, P. Roelse, B. Škorić, and B. de Weger, "Dynamic Tardos traitor tracing schemes," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4230–4242, 2013.

[5] G. Tardos, "Optimal probabilistic fingerprint codes," in *35th ACM Symposium on Theory of Computing (STOC)*, 2003, pp. 116–125.

[6] T. Laarhoven, J.-J. Oosterwijk, and J. Doumen, "Dynamic traitor tracing for arbitrary alphabets: Divide and conquer," in *IEEE Workshop on Information Forensics and Security (WIFS)*, 2012, pp. 240–245.

[7] J.-J. Oosterwijk, B. Škorić, and J. Doumen, "A capacity-achieving simple decoder for bias-based traitor tracing schemes," *Submitted*, 2013. [Online]. Available: http://eprint.iacr.org/2013/389

[8] Y.-W. Huang and P. Moulin, "On the saddle-point solution and the large-coalition asymptotics of fingerprinting games," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 160–175, 2012.



(a) The Tardos defense vs. Arbitrary attacks

(b) The interleaving defense vs. The interleaving attack

(c) The interleaving defense vs. The majority attack

(d) The interleaving defense vs. The minority attack

(e) The interleaving defense vs. The maj/minority attack

(f) The interleaving defense vs. The min/majority attack

Fig. 1. The dynamic interleaving defense in action against several attacks, together with one example of how the original dynamic Tardos scheme roughly works against arbitrary attacks. In this case, we used $c = 10$ and $n = 10^2$, $\varepsilon_1 = \varepsilon_2 = 10^{-3}$ as the simulation parameters, leading to $\ell \approx 14\,150$, $Z \approx 650$, and $\delta \approx 1/350$. The green area contains all innocent scores, the red lines show pirate scores, the bold purple line shows the average pirate score, and the bold blue curve shows the threshold $Z(i)$. The maj/minority attack consists of using the majority attack on the first 3000 positions and then switching to the minority attack, while in the min/majority attack they start with the minority attack and end with the majority attack. Note that the numbers on both axes are different in each figure, so both the number of segments needed to catch all colluders and the order of magnitude of user scores are different in each example.

[9] B. Škorić, S. Katzenbeisser, and M. Celik, "Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes," *Designs, Codes and Cryptography*, vol. 46, no. 2, pp. 137–166, 2008.

[10] T. Laarhoven and B. de Weger, "Discrete distributions in the Tardos scheme, revisited," in *1st ACM Workshop on Information Hiding and Multimedia Security (IHMMSec)*, 2013, pp. 13–18.

[11] O. Blayer and T. Tassa, "Improved versions of Tardos' fingerprinting scheme," *Designs, Codes and Cryptography*, vol. 48, no. 1, pp. 79–103, 2008.

[12] T. Laarhoven and B. de Weger, "Optimal symmetric Tardos traitor tracing schemes," *Designs, Codes and Cryptography*, 2012.

[13] B. Škorić and J.-J. Oosterwijk, "Binary and q-ary tardos codes, revisited," *Designs, Codes and Cryptography*, 2013.

[14] T. Furon, L. Pérez-Freire, A. Guyader, and F. Cérou, "Estimating the minimal length of Tardos code," in *11th Conference on Information Hiding (IH)*, 2009, pp. 176–190.

[15] A. Simone and B. Škorić, "Accusation probabilities in Tardos codes: Beyond the Gaussian approximation," *Designs, Codes and Cryptography*, vol. 63, no. 3, pp. 379–412, 2012.

[16] T. Furon and L. Pérez-Freire, "EM decoding of Tardos traitor tracing codes," in *11th ACM Symposium on Multimedia and Security (MMSec)*, 2009, pp. 99–106.

[17] T. Laarhoven, "Efficient probabilistic group testing based on traitor tracing," *51st Annual Allerton Conference on Communication, Control and Computing (Allerton)*, 2013.

[18] P. Meerwald and T. Furon, "Group testing meets traitor tracing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 4204–4207.