Analysis of the Non-Perfect Table Fuzzy Rainbow Tradeoff

Byoung-il Kim and Jin Hong

Department of Mathematical Sciences and ISaC,
Seoul National University, Seoul 151-747, Korea
{samaria2,jinhong}@snu.ac.kr

Abstract. Time memory tradeoff algorithms are tools for inverting one-way functions, and they are often used to recover passwords from unsalted password hashes. There are many publicly known tradeoff algorithms, and the rainbow tradeoff is widely believed to be the best algorithm. This work provides an accurate complexity analysis of the fuzzy rainbow tradeoff algorithm, which has not yet received much attention. It is shown that when the pre-computation cost and the online efficiency are both taken into consideration, the fuzzy rainbow tradeoff is preferable to the original rainbow tradeoff.

Keywords: time memory tradeoff, rainbow table, fuzzy rainbow, distinguished point

1 Introduction

Cryptanalytic time memory tradeoff algorithms are tools for quickly inverting one-way functions with the help of pre-computed data. They are used by law enforcement agencies and hackers to recover passwords from unsalted password hashes, and the multi-target variants of the algorithms have been used [7, 10, 11, 15] to show that the GSM mobile phones are insecure.

There are a multitude of publicly known tradeoff algorithms. However, a quick search for password recovery tools on the Web reveals [1–4] that the rainbow tradeoff [23] is by far the most popular algorithm, and this seems to indicate that the rainbow tradeoff is widely believed among implementers to be the best tradeoff algorithm.

Although it is difficult to clearly specify what is meant by one tradeoff algorithm to be better than another, the recent work [19] has given a plausible method for comparing the performances of different tradeoff algorithms. Unlike previous comparison attempts that had focused mainly on the optimal online phase behavior of the algorithms, the recently suggested method takes both the pre-computation cost and the online efficiency into account. Hence, the new method reflects our intuition concerning the practicality, usefulness, or value of the algorithms more closely. This approach was used in [19, 20] to show that the perfect and non-perfect rainbow tradeoffs perform better than the classical Hellman [16], perfect distinguished point [12–14], and non-perfect distinguished

point tradeoff algorithms, under typical environments, thus supporting the aforementioned beliefs.

In this work, we analyze the execution behavior of the non-perfect table fuzzy rainbow tradeoff [6, 8], which has not yet received much attention, and compare its performance against the original perfect and non-perfect rainbow tradeoffs. It is confirmed that the degree of online phase efficiency made possible by the original rainbow tradeoff through appropriate parameter choices is higher than that reachable with the fuzzy rainbow tradeoff. However, we find that, for online efficiency levels that can be reached by both tradeoff algorithms, the fuzzy rainbow tradeoff requires less pre-computation effort than the original rainbow tradeoff. In other words, up to a certain point, for the same pre-computation investment, higher online efficiency is returned by the fuzzy rainbow tradeoff.

Since the massive pre-computation requirement stands as a significant barrier to any large scale deployment of the tradeoff technique, the fuzzy rainbow tradeoff will often be preferable to the original rainbow tradeoff. The comparison given in this work follows the framework set by [19], and our main contribution is in providing an accurate execution behavior analysis of the non-perfect table fuzzy rainbow tradeoff.

The fuzzy rainbow tradeoff is a combination of the distinguished point tradeoff and the rainbow tradeoff. In fact, both the distinguished point and rainbow tradeoffs are special cases of the fuzzy rainbow tradeoff, corresponding to certain extreme parameter choices. Hence, one might expect the performance of the fuzzy rainbow tradeoff to come somewhere in between those of the distinguished point and rainbow tradeoffs. The findings of this paper, which indicate otherwise, could be seen as slightly unexpected.

Arguments supporting the efficiency of the fuzzy rainbow tradeoff were given in the original publications [6, 8]. However, they were based on the concept of hidden states, which totally disregarded pre-computation cost, and the computations made there were not tight enough to be accurate up to small constant factors. Since the performances of tradeoff algorithms often differ only by small constant factors, which nevertheless have heavy consequences in practice, it was not possible to provide an appropriate comparison of algorithms based on their claims.

The only work concerning the fuzzy rainbow tradeoff, other than the articles introducing the algorithm, that we are aware of is the presentation [21, 22] of a fully implemented attack on GSM phones. The tradeoff algorithm they explain as having used is the multi-target version of the fuzzy rainbow tradeoff. However, no theoretic analysis of the algorithm is given. We remark that the authors claim themselves to be the first in combining the distinguished point tradeoff and the rainbow tradeoff, hence it seems they were unaware of the preceding works [6, 8].

The rest of this paper is organized as follows. In Section 2, we review the fuzzy rainbow tradeoff algorithm. The subsequent three sections are devoted to analyzing the execution behavior of the fuzzy rainbow tradeoff. Section 3 gives the probability of success, Section 4 gives the accurate online time complexity that takes the effects of false alarms into account, and Section 5 discusses the

physical storage size required to record the pre-computation tables. After experimentally verifying the major parts of our arguments in Section 6, the theoretic results are used in Section 7 to present a fair comparison between the fuzzy rainbow tradeoff and the original rainbow tradeoff. Concluding remarks are made in Section 8.

2 Preliminaries

The reader is assumed to be familiar with the basic tradeoff techniques and the associated terminology.

Throughout this paper, the one-way function $f : \mathcal{N} \to \mathcal{N}$ is taken to acts on the search space $\mathcal{N}$ of size N. The composition of the one-way function and the reduction function of i-th color will be written as f_i. The standard notation for the number of chains m per table, the chain length t, and the number of tables ℓ will be used. When dealing with DPs (distinguished points), the distinguishing property will always be assumed to be of probability $\frac{1}{t}$, so that the expected length of a random chain is t. The collection of all chains associated with one pre-computation table is referred to as a pre-computation matrix.

The fuzzy rainbow tradeoff [6, 8] is a combination of the rainbow tradeoff and the DP tradeoff. Recall that the rainbow tradeoff uses length-t chains of the form

$$\mathrm{SP} \xrightarrow{f_1} \circ \xrightarrow{f_2} \circ \cdots \circ \xrightarrow{f_t} \mathrm{EP}, \tag{1}$$

where SP and EP denote the starting and ending points, respectively. Also recall that the DP tradeoff uses variable length chains of the form

$$\mathrm{SP} \xrightarrow{f_i} \circ \xrightarrow{f_i} \circ \cdots \circ \xrightarrow{f_i} \mathrm{DP} = \mathrm{EP}, \tag{2}$$

with a preset distinguishing property that defines DPs.

In the case of the fuzzy rainbow tradeoff, a distinguishing property and a positive integer s are fixed, and the chains of the form

$$\begin{aligned} \mathrm{SP} \xrightarrow{f_1} \circ \cdots \circ \xrightarrow{f_1} \mathrm{DP} \xrightarrow{f_2} \circ \cdots \circ \xrightarrow{f_2} \mathrm{DP} \xrightarrow{f_3} \circ \cdots \\ \cdots \xrightarrow{f_{s-1}} \mathrm{DP} \xrightarrow{f_s} \circ \cdots \circ \xrightarrow{f_s} \mathrm{DP} = \mathrm{EP} \end{aligned} \tag{3}$$

are used. That is, the iterations are continued under a fixed color until a DP is reached, after which the iterations are continued under a different color. A total of s colors are used for each pre-computation chain. As with other tradeoff algorithms, m chains are generated for each of the ℓ pre-computation tables, and only the starting point and ending point pairs, sorted according to the ending points, are stored in the pre-computation tables.

One can also consider the perfect tables version of this algorithm, obtained by retaining just one chain from every set of merging chains. However, only the non-perfect table version of the fuzzy rainbow tradeoff will be studied in this work.

The fuzzy rainbow tradeoff analogue of the matrix stopping rules is $mt^2s \approx \mathsf{N}$. In other words, we always assume that the parameters m, t, and s, for the fuzzy rainbow tradeoff, are chosen in such a way that the matrix stopping constant $\mathtt{F}_{\mathrm{msc}} = \frac{mt^2s}{\mathsf{N}}$ is neither too large nor very close to zero. We shall often express such a situation simply as $\mathtt{F}_{\mathrm{msc}} = \Theta(1)$. The appropriateness of this matrix stopping rule can be found in [6].

To complete the description of the fuzzy rainbow tradeoff algorithm, the order of online chain creation needs to be clarified. In short, the multiple tables are processed in parallel, analogous to the online phase of the rainbow tradeoff. In the initial pass, the online chains that start from the s-th color are generated for all the pre-computation tables. The second pass is executed only if the correct answer has not been found after the treatement of all alarms generated by the first pass online chains. In the second pass, the online chains that start from the $(s-1)$-th color and extends into the s-th color are generated for all the pre-computation tables. This process continues until either the correct answer is found or all s passes are complete.

The fuzzy rainbow tradeoff algorithm introduced by [6, 8] was a time memory data tradeoff algorithm that aimed to invert just one of multiple inversion targets. Since it is known [9] that the multi-target application of the rainbow tradeoff results in a tradeoff curve that is quite inferior to those of the multi-target classical Hellman or distinguished point tradeoffs, the intension of [6, 8] was to create a variant of the rainbow tradeoff with a multi-target tradeoff curve of $TM^2D^2 \approx \mathsf{N}^2$. However, in this work, we shall treat the fuzzy rainbow tradeoff as the single target inverting algorithm. The multi-target version of the fuzzy rainbow tradeoff must be compared against the multi-target versions of the classical Hellman and DP tradeoffs, but nothing similar to [19, 20] has yet appear for these algorithms. Furthermore, preliminary analysis seem to indicate that the analyses for the single target inversion algorithms will carry over to the multi-target algorithms almost word for word.

The fuzzy rainbow matrix may be viewed as a concatenation of s DP sub-matrices, with the ending points of one DP sub-matrix used as the starting points for the next DP sub-matrix. Throughout this work, the i-th ($i = 1, \ldots, s$) DP sub-matrix will be denoted by $\mathtt{DM}_i$. The only difference between $\mathtt{DM}_i$ and a normal non-perfect DP matrix is that $\mathtt{DM}_i$ may contain duplicate starting points, that bring about fully identical chains.

Any implementation of a tradeoff algorithm that relies on DPs will set a chain length bound to detect chains falling into loops. In this work, we assume that a sufficiently large chain length bound is used. This is not equivalent to sending the chain length bound to infinity during analysis. A more detailed discussion of the exact meaning of this assumption may be found in [19, 20].

There will be many approximations made throughout this paper. Most of these will depend on the relation $(1 - \frac{1}{b})^a \approx e^{-\frac{a}{b}}$, which is appropriate when $a = O(b)$. A more precise statement can be found in [19]. Under any reasonable choice of tradeoff parameters, these approximations will be very accurate whenever we apply the relation, and they will be written as equalities rather

than as approximations. Another class of approximations appearing in this paper will involve interpretation of finite sums as definite integrals. Once again, when the summation is made over a large index set so that the approximation is accurate, we shall silently write the relation as an equality rather than as an approximation.

3 Probability of Success

The one-way function is assumed to be a random function throughout the analysis of this paper.

The number of one-way function invocations required to construct all the pre-computation tables is expected to be $mts\ell$. Let us define the *pre-computation coefficient* of the fuzzy rainbow tradeoff that uses parameters m, t, s, and ℓ to be

$$\mathtt{F}_{\mathrm{pc}} = \frac{mts\ell}{\mathsf{N}}, \tag{4}$$

so that, when the much smaller $m\ell \log m$ order effort of sorting is ignored, we may state $\mathtt{F}_{\mathrm{pc}}\mathsf{N}$ as the cost of pre-computation.

We also define the *coverage rate* of a fuzzy rainbow matrix to be

$$\mathtt{F}_{\mathrm{cr}} = \frac{1}{mts}\big(|\mathtt{DM}_1| + |\mathtt{DM}_2| + \cdots + |\mathtt{DM}_s|\big), \tag{5}$$

where $|\mathtt{DM}_i|$ denotes the number of distinct points expected in the i-th DP sub-matrix. Readers familiar with the definitions of the coverage rate appearing in previous works should note the slight difference. The current definition does not refer to the total number of distinct points in the pre-computation matrix. However, the following proposition shows that this is the natural definition to use in the case of fuzzy rainbow tradeoffs.

Proposition 1. *Given an inversion target created from a random input to the one-way function, the fuzzy rainbow tradeoff will succeed in recovering the correct input with probability*

$$\mathtt{F}_{\mathrm{ps}} = 1 - e^{-\mathtt{F}_{\mathrm{cr}}\mathtt{F}_{\mathrm{pc}}}.$$

Proof. Since the probability of success expected from one DP sub-matrix $\mathtt{DM}_i$ is $\frac{|\mathtt{DM}_i|}{\mathsf{N}}$, treating the DP sub-matrices that were created with different reduction functions as being independent, we can write the success rate of the complete online phase as

$$\mathtt{F}_{\mathrm{ps}} = 1 - \prod_{i=1}^{s}\Big(1 - \frac{|\mathtt{DM}_i|}{\mathsf{N}}\Big)^{\ell}.$$

This may be approximated by

$$\begin{aligned}\mathtt{F}_{\mathrm{ps}} = 1 - \prod_{i=1}^{s}\exp\Big(-\frac{|\mathtt{DM}_i|\ell}{\mathsf{N}}\Big) &= 1 - \exp\Big(-\sum_{i=1}^{s}|\mathtt{DM}_i|\frac{\ell}{\mathsf{N}}\Big)\\ &= 1 - \exp\Big(-\mathtt{F}_{\mathrm{cr}}\frac{mts\ell}{\mathsf{N}}\Big) = 1 - \exp\big(-\mathtt{F}_{\mathrm{cr}}\mathtt{F}_{\mathrm{pc}}\big),\end{aligned}$$

as claimed. □

This proposition does not regard the finding of a pre-image that is different from the input used to create the inversion target as being successful. Also, it does not deal with the situation where the inversion target, rather than the one-way function input, is chosen at random. These different versions of the inversion problem behave sufficiently differently to require separate analyses. A more careful definition of $|\mathtt{DM}_i|$, suitable for the inversion problem considered in this work, would have counted only the points that were used as inputs to the one-way function during the pre-computation, disregarding the ending points. A more detailed treatment of these technical details, which we do not illustrate fully in the remained of this paper, can be found in [19].

To utilize the above proposition, we need a way to express the coverage rate in terms of the tradeoff algorithm parameters. Recall from [18, 20] that the number of distinct entries $|\mathtt{DM}|$ contained in a non-perfect DP matrix, excluding the ending points, satisfies

$$|\mathtt{DM}| = m_{\mathrm{ep}} t, \tag{6}$$

where m_{ep} denotes the number of distinct ending points of the DP matrix. The work [20] uses this fact in conjunction with another expression for $|\mathtt{DM}|$, found in [19], to obtain the formula

$$m_{\mathrm{ep}} = m_{\mathrm{sp}} \frac{2}{1 + \sqrt{1 + \frac{2 m_{\mathrm{sp}} t^2}{\mathtt{N}}}}, \tag{7}$$

that relates the number of distinct starting points m_{sp} to the number of distinct ending points m_{ep} in a DP matrix.

Returning to the fuzzy rainbow matrices, let us use m_{i-1} and m_i to denote the number of distinct starting points and ending points, respectively, expected in each DP sub-matrix $\mathtt{DM}_i$. In particular, $m_0 = m$ and m_s are the numbers of distinct starting and ending points, respectively, of the full fuzzy rainbow matrix. We shall refer to each collection of m_i points as the *i-th color boundary points* of a fuzzy rainbow matrix, where $i = 0, 1, \ldots, s$. Adopting the above two facts to our situation, we can state that

$$|\mathtt{DM}_i| = m_i t \tag{8}$$

and

$$m_{i+1} = m_i \frac{2}{1 + \sqrt{1 + \frac{2 m_i t^2}{\mathtt{N}}}} \quad \text{with} \quad m_0 = m \tag{9}$$

are to be expected at each applicable color index i. The following closed-form formula for m_i is easier to utilize than the iterative formula (9).

Lemma 1. *When the number of colors s used in each pre-computation table is large, the number of i-th color boundary points in a fuzzy rainbow matrix is expected to be*

$$m_i = \frac{2m}{2 + \mathtt{F}_{\mathrm{msc}} \frac{i}{s}},$$

for $i = 0, 1, \ldots, s$.

Proof. The iterative formula (9), written in a series expansion form, is

$$m_{i+1} = m_i\Big(1 - \frac{1}{2}\frac{m_i t^2}{\mathsf{N}}\Big) + O\Big(m_i\Big(\frac{m_i t^2}{\mathsf{N}}\Big)^2\Big).$$

Since the lemma statement assumes $\frac{m_i t^2}{\mathsf{N}} = O\big(\frac{1}{s}\big)$ to be small, we can ignore the big-O part and rewrite this as the difference equation

$$\frac{m_{i+1}}{m} - \frac{m_i}{m} = -\frac{\mathtt{F}_{\mathrm{msc}}}{2s}\Big(\frac{m_i}{m}\Big)^2.$$

As an application of the Euler method, we can interpret this as the differential equation

$$y' = -\frac{\mathtt{F}_{\mathrm{msc}}}{2s}y^2$$

with the initial condition $y(0) = \frac{m_0}{m} = 1$, and solve for y to obtain

$$\frac{m_i}{m} = \frac{2}{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}},$$

which is the claimed statement. □

The coverage rate, at least for the large s values, follows directly from this lemma.

Proposition 2. *When the number of colors s used in each pre-computation table is large, the coverage rate of a fuzzy rainbow matrix is*

$$\mathtt{F}_{\mathrm{cr}} = \frac{2}{\mathtt{F}_{\mathrm{msc}}}\ln\Big(1 + \frac{\mathtt{F}_{\mathrm{msc}}}{2}\Big).$$

Proof. Combining Lemma 1 and (8), one can check that

$$\mathtt{F}_{\mathrm{cr}} = \frac{1}{mts}\sum_{k=1}^{s}|\mathtt{DM}_k| = \frac{1}{s}\sum_{k=1}^{s}\frac{m_k}{m} = \sum_{k=1}^{s}\frac{2}{2 + \mathtt{F}_{\mathrm{msc}}\frac{k}{s}}\frac{1}{s} = \int_0^1 \frac{2}{2 + \mathtt{F}_{\mathrm{msc}}u}\,du.$$

The final definite integral can be computed to confirm what is claimed. □

The computation done in this proof also gives the partial coverage rate

$$\frac{1}{mts}\sum_{k=i+1}^{s}|\mathtt{DM}_k| = \frac{1}{s}\sum_{k=i+1}^{s}\frac{m_k}{m} = \frac{2}{\mathtt{F}_{\mathrm{msc}}}\ln\Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}}{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}\Big), \tag{10}$$

which contains Proposition 2 as the special case of $i = 0$.

The arguments given so far supports the validity of formula (10) only for large s values. There are two sources of inaccuracy, the first being the Euler method used in Lemma 1 and the second being the interpretation of a summation

an a definite integral. However, one can verify through explicit computations that the claim (10) is accurate even when s is very small.

After rewriting (9) in the form

$$\frac{m_{i+1}}{m} = \frac{m_i}{m} \frac{2}{1+\sqrt{1+2\,\frac{\mathtt{F}_{\mathrm{msc}}}{s}\frac{m_i}{m}}} \quad \text{and} \quad \frac{m_0}{m} = 1, \tag{11}$$

one can iteratively compute all $\frac{m_i}{m}$, for any given $\mathtt{F}_{\mathrm{msc}}$ and small s. In Figure 1,

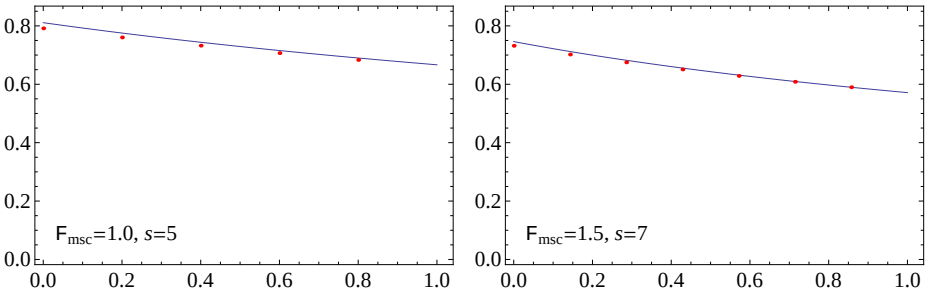

Fig. 1. Comparisons between closed-form formula (*line*) and iteratively computed values (*dots*) of the partial coverage rate at small s. (x-axis: $\frac{i}{s}$; y-axis: $\frac{s}{s-i}\sum_{k=i+1}^{s}\frac{\lfloor \mathtt{DM}_k \rfloor}{mts}$)

we have compared

$$\frac{s}{s-i} \times \frac{1}{s} \sum_{k=i+1}^{s} \frac{m_k}{m}, \tag{12}$$

with $\frac{m_k}{m}$ iteratively computed through (11), against the graphs of

$$\frac{s}{s-i} \times \frac{2}{\mathtt{F}_{\mathrm{msc}}} \ln\left(\frac{2+\mathtt{F}_{\mathrm{msc}}}{2+\mathtt{F}_{\mathrm{msc}}\frac{i}{s}}\right), \tag{13}$$

at very small s values. The $\frac{s}{s-i}$ factor has been multiplied to the partial coverage rates so that all values are close to 1 and they may be gather within a single figure. The figures testify that Lemma 1, Proposition 2, and formula (10) are accurate even at these very small s values. Since it will later become clear that the s values of interest will be somewhat larger than those given by these explicitly computed examples, henceforth, we shall treat Lemma 1, Proposition 2, and formula (10) as being valid for all s values of interest.

Remark 1. Proposition 1 implies that any set of parameters m, t, s, and ℓ that achieves the success rate $\mathtt{F}_{\mathrm{ps}}$ must satisfy the relation

$$\frac{\ell}{t} = \frac{\mathtt{F}_{\mathrm{pc}}}{\mathtt{F}_{\mathrm{msc}}} = \frac{\{-\ln(1-\mathtt{F}_{\mathrm{ps}})\}}{\mathtt{F}_{\mathrm{msc}}\mathtt{F}_{\mathrm{cr}}}, \tag{14}$$

and Proposition 2 gives the coverage rate $\mathtt{F}_{\mathrm{cr}}$ as a function of the single variable $\mathtt{F}_{\mathrm{msc}}$. Hence, when parameter sets are restricted to those that achieve a fixed requirement $\mathtt{F}_{\mathrm{ps}}$ on the success rate, the ratio $\frac{\ell}{t}$ may also be seen as a function of the single variable $\mathtt{F}_{\mathrm{msc}}$.

Remark 2. Recall that we are working with parameters for which $\mathtt{F}_{\mathrm{msc}}$ is of $\Theta(1)$ order. This fact and Proposition 2 implies that the coverage rate

$$\mathtt{F}_{\mathrm{cr}} = 1 - \frac{1}{2}\Big(\frac{\mathtt{F}_{\mathrm{msc}}}{2}\Big) + \frac{1}{3}\Big(\frac{\mathtt{F}_{\mathrm{msc}}}{2}\Big)^2 - \frac{1}{4}\Big(\frac{\mathtt{F}_{\mathrm{msc}}}{2}\Big)^3 + \cdots$$

is also of $\Theta(1)$ order. Hence, unless the success rate requirement $\mathtt{F}_{\mathrm{ps}}$ is unrealistically close to 100%, relation (14) implies that ℓ and t are of similar order.

4 Online Complexity

Having secured full knowledge concerning the success rate of the fuzzy rainbow tradeoff, we next discuss the online execution complexities. We start with the cost of generating the online chains. Note that our interest is in the average case complexity, rather than the worst case complexity.

We first need to obtain the probability for each pass of the online phase to be executed.

Lemma 2. *The probability for the online chains that start from the i-th color of each fuzzy rainbow matrix to be generated, i.e., the probability for the DP sub-matrix $\mathtt{DM}_i$ within each fuzzy rainbow matrix to be searched for the correct answer to the inversion target, is*

$$\Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}}.$$

Proof. The online chains that start from the i-th color of each fuzzy rainbow matrix will be generated if and only if the correct answer to the inversion target does not belong to the DP sub-matrices $\mathtt{DM}_{i+1}, \ldots, \mathtt{DM}_s$ contained in the ℓ fuzzy rainbow matrices. Hence, the probability under consideration is

$$\prod_{k=i+1}^{s}\Big(1 - \frac{|\mathtt{DM}_k|}{\mathtt{N}}\Big)^{\ell} = \exp\Big(-\frac{\ell}{t}\mathtt{F}_{\mathrm{msc}}\sum_{k=i+1}^{s}\frac{|\mathtt{DM}_k|}{mts}\Big) = \Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}},$$

where the second equality follows from (10). □

The following statement is a direct consequence of this lemma.

Proposition 3. *The generation of the online chains during the online phase of a fuzzy rainbow tradeoff is expected to require*

$$t\ell\sum_{i=1}^{s}(s - i + 1)\Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}}$$

iterations of the one-way function.

Proof. Each of the ℓ online chains that start from the i-th color of each fuzzy rainbow matrix is expected to require $(s-i+1)t$ iterations of the one-way function. The probability for each of these iterations to be executed is given by Lemma 2. The claim is a simple combination of these two observations. □

Our next objective is to express the cost of dealing with alarms, which is the remaining component of the online time complexity. To compute the cost of resolving alarms associated with an online chain that starts from the i-th color, the merging of single colored DP chains needs to be considered first.

Lemma 3. *The probability for two randomly generated DP chains to merge into each other is* $\frac{t^2}{\mathsf{N}}$.

Proof. The probability for the first chain to become a DP chain of length i is $\big(1-\frac{1}{t}\big)^{i-1}\frac{1}{t}$. The probability for the second chain to merge into this chain at its j-th iteration is $\big(1-\frac{1}{t}-\frac{i+1}{\mathsf{N}}\big)^{j-1}\frac{i+1}{\mathsf{N}}$. Hence, the probability for two chains to merge may be expressed as

$$\sum_{i=1}^{\infty}\big(1-\frac{1}{t}\big)^{i-1}\frac{1}{t}\sum_{j=1}^{\infty}\big(1-\frac{1}{t}-\frac{i+1}{\mathsf{N}}\big)^{j-1}\frac{i+1}{\mathsf{N}},$$

where the two infinite sums should be understood as expressing the summations up to a sufficiently large chain length bound, so that we have $\frac{i+1}{\mathsf{N}} \ll \frac{1}{t}$ and $\big(1-\frac{1}{t}-\frac{i+1}{\mathsf{N}}\big)^{j-1} \approx \big(1-\frac{1}{t}\big)^{j-1}$. The above may now be approximated by

$$\frac{t^2}{\mathsf{N}}\int_0^{\infty}\int_0^{\infty} e^{-u}e^{-v}u\,dv\,du = \frac{t^2}{\mathsf{N}},$$

and this completes the proof. □

With the simplifying assumption that the two DP chains are of length t, the merge probability can be computed more simply as

$$1-\Big(1-\frac{t}{\mathsf{N}}\Big)^t = \frac{t^2}{\mathsf{N}} - \binom{t}{2}\Big(\frac{t}{\mathsf{N}}\Big)^2 + \cdots - (-1)^t\binom{t}{t}\Big(\frac{t}{\mathsf{N}}\Big)^t \approx \frac{t^2}{\mathsf{N}}, \qquad (15)$$

which is in agreement with the lemma. This approach seems trivial enough to have been previously known by many researcher. We had treated the chain lengths carefully in the proof, since merge probabilities are higher with longer chains, and it was unclear as to whether this fact could bring about a difference in the final expression.

The single color merge probability is now used to compute the cost of resolving alarms from the fuzzy rainbow chains.

Lemma 4. *Consider a single fuzzy rainbow pre-computation matrix and its associated s colors. Assume the generation of an online chain for this matrix that starts from the i-th color. The cost of resolving alarms that may be induced by*

possible merges of this online chain with the chains of the fuzzy rainbow matrix is expected to be

$$t\frac{\mathsf{F}_{\mathrm{msc}}}{s}\big\{i(s-i+1)+1\big\}$$

iterations of the one-way function.

Proof. The alarms need to be handled separately for each pre-computation chain that merges with the online chain. The resolving of an alarm associated with one pre-computation chain requires the same amount of work regardless of whether or not the pre-computation and online chains merge with other pre-computation chains. Hence, the total cost of resolving alarms is m times the cost associated with one pre-computation chain. Below, we will focus on the possible merge between the online chain and a single randomly generated pre-computation chain.

If the online chain that starts from the i-th color merges with a given pre-computation chain, the merge will be at precisely one of the colors i, $i+1$, …, s. The case of possible merge at the i-th color will be treated separately from the possible merge at strictly later colors.

The probability for the i-th colored DP sub-chain of the pre-computation chain to be of length p is $\big(1-\frac{1}{t}\big)^{p-1}\frac{1}{t}$. The probability for the online chain to merge into this DP sub-chain of length p within its i-th color DP sub-chain is

$$\sum_{q=1}^{\infty}\Big(1-\frac{1}{t}-\frac{p+1}{\mathsf{N}}\Big)^{q-1}\frac{p+1}{\mathsf{N}}=\frac{p+1}{\mathsf{N}}\frac{1}{\frac{1}{t}+\frac{p+1}{\mathsf{N}}}\approx\frac{t(p+1)}{\mathsf{N}}.$$

To resolve the alarm from such a merge, one must expect to compute $(i-1)t$ iterations of the one-way function to regenerate the pre-computation chain up to the start of the i-th colored sub-chain and then additionally compute p iterations to reach the end of the i-th colored sub-chain. The only exception is when the correct answer is found, but this is a single rare event among the many alarms, and can be ignored. Hence, the cost of resolving an alarm that may occur due to a possible merge at the i-th color is

$$\sum_{p=1}^{\infty}\Big(1-\frac{1}{t}\Big)^{p-1}\frac{1}{t}\cdot\frac{t(p+1)}{\mathsf{N}}\cdot\big\{(i-1)t+p\big\},$$

which may be approximated by

$$\frac{t^3}{\mathsf{N}}\int_0^{\infty}e^{-u}u\big\{(i-1)+u\big\}\,du=(i+1)\frac{t^3}{\mathsf{N}}.$$

For color indices $j>i$, we can infer from Lemma 3 that the merge of the two chains at the j-th color will occur with probability $\big(1-\frac{t^2}{\mathsf{N}}\big)^{j-i}\frac{t^2}{\mathsf{N}}$. The combined probability of merge at any one of the colors appearing strictly after the i-th color is

$$\sum_{j=i+1}^{s}\Big(1-\frac{t^2}{\mathsf{N}}\Big)^{j-i}\frac{t^2}{\mathsf{N}}=\Big(1-\frac{t^2}{\mathsf{N}}\Big)\Big\{1-\Big(1-\frac{t^2}{\mathsf{N}}\Big)^{s-i}\Big\}\approx(s-i)\frac{t^2}{\mathsf{N}}.$$

The final approximation is true since $\frac{t^2}{\mathsf{N}} \ll 1$, and the same result would have been obtained if we had simply treated the merge probability at each color to be $\frac{t^2}{\mathsf{N}}$, regardless of j. Since any such merge will require just it iterations of the one-way function to resolve, the sum of costs associated with possible merges at all colors appearing strictly after the i-th color is

$$(s-i)\frac{t^2}{\mathsf{N}} \cdot it.$$

The sum of the two computed costs

$$(i+1)\frac{t^3}{\mathsf{N}} + i(s-i)\frac{t^3}{\mathsf{N}} = \big\{i(s-i+1)+1\big\}\frac{t^3}{\mathsf{N}}$$

is the cost expected from a single pre-computation chain, and m times this value is what is claimed. □

In the case of the DP tradeoff, one can reduce [18] the cost of resolving alarms slightly through the online chain record (OCR) technique [17, 24]. The method is to keep a full record of the online chain during its generation, so that any pre-computation chain regeneration can be stopped at the exact point of merge, rather than at the terminating DP. Some readers may have wondered why the OCR technique is not being applied to the fuzzy rainbow tradeoff. A careful reading of the above proof shows that, for an online chain that starts from the i-th color, the OCR technique can take effect only if the merge occurs within the i-th colored sub-chain. This observation indicates that the application of OCR technique to the fuzzy rainbow tradeoff will reduce the number of one-way function iterations, at the cost of more frequent memory accesses, but the reduction will be too minimal to be of interest.

During the online phase, the generation of the online chain that starts from the i-th color, which the above lemma assumes, is done only if all previous shorter online chains have failed to return the correct answer. The following statement accounts for this in computing the cost of resolving alarms.

Proposition 4. *The resolving of alarms during the online phase of a fuzzy rainbow tradeoff is expected to require*

$$t\ell\,\frac{\mathtt{F}_{\mathrm{msc}}}{s}\sum_{i=1}^{s}\big\{i(s-i+1)+1\big\}\Big(\frac{2+\mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2+\mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}}$$

iterations of the one-way function.

Proof. The probability for the online chains that start from the i-th color of each pre-computation matrix to be generated is given by Lemma 2. The cost of alarm treatment expected from each of these online chains is stated by Lemma 4. It now suffices to multiply by ℓ to account for the multiple online chains and sum the product of the mentioned probability and expected work factor over all color indices. □

The two components of the online time complexity for the fuzzy rainbow tradeoff have been obtained, and we are ready to state the tradeoff coefficient, which succinctly expresses the online efficiency of a tradeoff algorithm.

Theorem 1. *The time memory tradeoff curve for the non-perfect table fuzzy rainbow tradeoff that uses s colors per pre-computation matrix is $TM^2 = \mathtt{F}_{\mathrm{tc},s}\mathsf{N}^2$, where the tradeoff coefficient is*

$$\mathtt{F}_{\mathrm{tc},s} = \mathtt{F}_{\mathrm{msc}}^2\Big(\frac{\ell}{t}\Big)^3 \sum_{i=1}^{s} \left(\Big(1 - \frac{i-1}{s}\Big)\Big(1 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}\Big) + \frac{\mathtt{F}_{\mathrm{msc}}}{s^2} \right) \Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}} \frac{1}{s}.$$

Proof. The storage complexity of the fuzzy rainbow tradeoff is $M = m\ell$, and the time complexity is the sum

$$T = t\ell \sum_{i=1}^{s} \Big\{ (s-i+1)\Big(\mathtt{F}_{\mathrm{msc}}\frac{i}{s} + 1\Big) + \frac{\mathtt{F}_{\mathrm{msc}}}{s} \Big\} \Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}}$$

of the two terms given by Proposition 3 and Proposition 4. The tradeoff curve is obtained by appropriately combining the two complexities M and T. Easy modifications of the formula are needed to arrive at the form of the tradeoff coefficient presented by this theorem. □

Before ending this section, we state the number of table lookups expected during the online phase. The time required for table lookups is implementation dependent. Since, when the pre-computation tables reside in fast memory, the cost of table lookups could be negligible in comparison to the cost of one-way function computations, table lookups are mostly ignored in theoretic analyses of the tradeoff algorithms. The same approach will be taken by this paper and the result below will not be required in the rest of this paper. In practice, table lookups can become a bottleneck and affect the performance of the algorithms significantly.

Proposition 5. *The online phase of the fuzzy rainbow tradeoff is expected to call for*

$$\ell \sum_{i=1}^{s} \Big(\frac{2 + \mathtt{F}_{\mathrm{msc}}\frac{i}{s}}{2 + \mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}},$$

table lookups.

Proof. The probability for the online chains that start from the i-th color of each pre-computation matrix to be generated is given by Lemma 2, and every such online chain generation will call for a single table lookup per pre-computation table. Hence, the expected number of lookups can be written as claimed. □

Note that the above table lookup count is upper bounded by $s\ell$ and lower bounded by $s\ell\big(\frac{2}{2+\mathtt{F}_{\mathrm{msc}}}\big)^{2\frac{\ell}{t}}$. Recalling Remark 2, given at the end of Section 3, we may state that the number of table lookups made by the online phase of the fuzzy rainbow tradeoff is of $\Theta(ts)$ order.

Later discussions in Section 7.3 show that ts for the fuzzy rainbow tradeoff corresponds naturally to t for the original rainbow tradeoff, in a manner that is somewhat analogous to how mt for the classical Hellman tradeoff corresponds to m for the rainbow tradeoff. Since the original rainbow tradeoff requires $\Theta(t\ell) = \Theta(t)$ table lookups [19], the table lookup requirements of the fuzzy rainbow and original rainbow tradeoffs are comparable.

5 Storage Optimization

The storage complexity M appearing in the tradeoff curve of Theorem 1 refers to the number of entries, i.e., starting and ending point pairs, that are written to the pre-computation tables. In practice, the physical storage size, which depends not only on the number of table entries, but also on how many bits of storage must be allocated to each table entry, will be more important.

Each randomly generated starting point requires $\log \mathsf{N}$ bits of storage to record. However, notice that, as long as we are treating the one-way function as a random function, any set of m distinct starting points is as good as a set of randomly generated starting points. If sequentially generated starting points [5, 11, 12] are used, each starting point can be stored in $\log m$ bits, which is usually much smaller than the original $\log \mathsf{N}$ bits.

The issue of recording the ending points effectively is more complicated. There are three major techniques that can be used with various tradeoff algorithms. Slightly more detail than what is explained below can be found in [19, 20].

The first of the three methods is the index file method [11], which is widely used even outside the tradeoff subject. Once a pre-computation table has been sorted according to the ending points, two consecutive ending points in the table are highly likely to share a small number of common significant bits. This predictability of the significant bits can be used to remove almost $\log m$ bits per table entry, without any loss of information concerning the ending points, when the table contains m entries.

The second method for reducing ending point storage size is applicable only when the ending points are DPs. One simply does not have to record any portion of the ending points that can be recovered from the distinguishing property [11, 25]. This allows removal of $\log t$ bits from each ending point without any information loss, when the distinguishing property is of $\frac{1}{t}$ probability. Clearly, the fuzzy rainbow tradeoff allows application of this technique.

The final technique is to simply truncate the ending points to a certain length before recording them to storage [8, 11]. During the online phase, the terminating DP of an online chain is likewise truncated before being searched for in the pre-computation table. This reduces the storage requirement, but since partial matches of ending points will now be incorrectly announced as collisions, this will cause a new type of false alarms to appear and increase the cost of resolving alarms. In the remainder of this section we work to find the degree of truncation that restricts the side effects of ending point truncation to a negligible fraction of the online time complexity.

Let us assume a fixed truncation method for ending points with a truncated match probability of $\frac{1}{r}$. That is, we assume that the truncated outcome of two independently and randomly chosen ending points, which are a priori DPs, will be identical with probability $\frac{1}{r}$. For example, the case of no truncation corresponds to $r = \frac{\mathsf{N}}{t}$. In most cases, the truncated match probability of $\frac{1}{r}$ can be obtained by retaining just $\log r$ bits of each ending point, that are unrelated to the distinguishing property. Note that whether the part that can be recovered from the distinguishing property, which contains no entropy, is also retained, does not affect the truncated match probability.

The extra cost incurred by the truncation-related alarms is stated below.

Proposition 6. *Assume the use of the ending point truncation method with the truncated match probability set to $\frac{1}{r}$. Then, during the online phase of the fuzzy rainbow tradeoff, one can expected to observe*

$$t\ell \frac{m}{r} \sum_{i=1}^{s} i \left(\frac{2 + \mathsf{F}_{\mathrm{msc}} \frac{i}{s}}{2 + \mathsf{F}_{\mathrm{msc}}} \right)^{2\frac{\ell}{t}}$$

extra invocations of the one-way function induced by truncation-related alarms.

Proof. As was argued during the proof of Lemma 4, it suffices to focus on the possible merges between a set of s online chains starting at different colors and a single pre-computation chain, and later account for multiple pre-computation chains.

We first need to separate the normal alarms from alarms caused by ending point truncations. Consider an online chain that starts from the i-th color. Through an argument similar to that appearing in the proof of Lemma 4, we can deduce from Lemma 3 that the probability for the online chain *not* to merge into any single fixed pre-computation is $1 - (s - i + 1)\frac{t^2}{\mathsf{N}}$. Unless r is close to $\frac{\mathsf{N}}{t}$, the probability for the non-merging two chains to bring about a truncated match is $\frac{1}{r}$. Hence, the probability for an online chain that starts from the i-th color to cause a truncation related alarm is

$$\left\{1 - (s - i + 1)\frac{t^2}{\mathsf{N}}\right\}\frac{1}{r}.$$

Each of these pseudo-alarms will require it iterations of the one-way function to resolve. Taking the ℓm pre-computation chains into account and recalling Lemma 2, which gives the probability for an online chain that starts from the i-th color to be generated, the cost of dealing with truncation-related alarms can be written as

$$\ell m \sum_{i=1}^{s} it \cdot \left\{1 - (s - i + 1)\frac{t^2}{\mathsf{N}}\right\}\frac{1}{r} \cdot \left(\frac{2 + \mathsf{F}_{\mathrm{msc}} \frac{i}{s}}{2 + \mathsf{F}_{\mathrm{msc}}} \right)^{2\frac{\ell}{t}}.$$

It now suffices to observe that $(s - i + 1)\frac{t^2}{\mathsf{N}} = O\left(\frac{1}{m}\right)$ to realize that the claimed formula is an accurate approximation. □

The cost stated by this proposition is upper bounded by

$$t\ell\,\frac{m}{r}\sum_{i=1}^{s} i\left(\frac{2+\mathtt{F}_{\mathrm{msc}}\frac{s}{s}}{2+\mathtt{F}_{\mathrm{msc}}}\right)^{2\frac{\ell}{t}} = t\ell\frac{m}{r}\,\frac{s(s+1)}{2}$$

and lower bounded by

$$t\ell\,\frac{m}{r}\sum_{i=1}^{s} i\left(\frac{2+\mathtt{F}_{\mathrm{msc}}\frac{0}{s}}{2+\mathtt{F}_{\mathrm{msc}}}\right)^{2\frac{\ell}{t}} = t\ell\frac{m}{r}\,\frac{s(s+1)}{2}\big(\frac{2}{2+\mathtt{F}_{\mathrm{msc}}}\big)^{2\frac{\ell}{t}}.$$

Recalling Remark 2, given at the end of Section 3, we can state that the added cost of dealing with truncation-related alarms is of $\Theta\big(t^2s^2\frac{m}{r}\big)$ order.

On the other hand, the time complexity T, appearing in the proof of Theorem 1, can be bounded from above and below by

$$t\ell\frac{s(s+1)}{2}\Big(\frac{2}{2+\mathtt{F}_{\mathrm{msc}}}\Big)^{2\frac{\ell}{t}} \le T \le t\ell\Big\{\frac{s(s+1)}{2}(\mathtt{F}_{\mathrm{msc}}+1)+\mathtt{F}_{\mathrm{msc}}\Big\},$$

and this implies that $T=\Theta(t^2s^2)$.

Comparison of the two time complexities orders $\Theta\big(t^2s^2\frac{m}{r}\big)$ and $\Theta(t^2s^2)$ implies that, if $\frac{m}{r}$ is a sufficiently small fraction, then the added cost of treating truncation-related alarms will be insignificant in comparison to the time complexity T for the algorithm that does not use truncation of ending points. In other words, the side effects of ending point truncation can be ignored if the truncated ending points contain slightly more than $\log m$ bits of information. Of course, if the truncated ending points still contain bits that can be recovered from the DP definition they may also be removed without any loss of information. Furthermore, even the remaining effective $\log m$ bits of the ending point can mostly be removed through the index table method, without any loss of information.

In summary, storage of each starting point of the fuzzy rainbow tradeoff requires $\log m$ bits and the storage of each ending point requires a very small number ε of bits. Each entry of the fuzzy rainbow tradeoff can be recorded in $\log m+\varepsilon$ bits.

Now that we have seen the detailed effects of the ending point truncation method, we can present an overall interpretation of the inner workings that is easier to understand. Note that, not only the total time complexity T, but also the cost of resolving alarms is of $\Theta(t^2s^2)$ order, and that each alarm induces $\Theta(ts)$ iterations of the one-way function, on average. Hence, one is expected to encounter $\Theta(ts)$ alarms, or merges between an online chain and a pre-computation chain, during the online phase. Since this is equal to $\Theta(s\ell)$, the number of table lookups expected during the online phase, one can conclude that each table lookup incurs $\Theta(1)$ alarm, on average. Hence, as long as slightly more than $\log m$ bits of each ending point are retained, the ending points within any single pre-computation table will remain distinguishable from each other, and the number of pseudo-collisions will be kept at a small fraction of the number of true merges.

6 Experimental Results

Test results that support our theoretic findings are given in this section.

The one-way function used during all the tests was the key to ciphertext mapping, under a randomly generated fixed plaintext, of AES-128. Freshly generated random plaintexts were used each time we required a new random mapping for multiple independent tests. Truncation of ciphertexts to 40 bits and zero-extension to 128-bit keys were used to control the search space to a manageable size of $\mathsf{N} = 2^{40}$. The parameter t was always taken to be an integer power of 2, and the distinguishing property was set to check whether $\log t$ least significant bits were zero. XOR-ing by constants were used as the reduction functions, with the binary expression of i used as the i-th color constant.

Our first experiment verifies Lemma 1. This should give strength to the validity of the partial coverage rate formula (10). After fixing parameters m, t, s, and N, we began the matrix generation with m starting points. Pre-computation chains were initially generated only up to the first set of DPs, i.e., only the first DP sub-matrix $\mathtt{DM}_1$ was initially generated. After sorting these color boundary DPs, any duplicates were discarded, and the number of remaining set of distinct DPs was recorded. The second DP sub-matrix was then generated from the distinct set of DPs, and the number of second group of color boundary DPs was recorded. The process was continued for s colors.

A small number of the chains did not reach a DP within our chain length bound of $15t$ at various colors and were prematurely discarded. A total of ten fuzzy rainbow matrices were generated, and the number of i-th color boundary points was averaged separately for each i.

Table 1. Intermediate DP counts $\frac{m_i}{m}$ in a fuzzy rainbow matrix for a small s. ($m = 2000$, $t = 2^{13}$, $s = 7$, $\mathsf{N} = 2^{40}$)

i	0	1	2	3	4	5	6	7
test	1.0000	0.9457	0.8955	0.8494	0.8094	0.7743	0.7419	0.7119
Eq. (11)	1.0000	0.9454	0.8964	0.8521	0.8119	0.7752	0.7416	0.7108
Lemma 1	1.0000	0.9425	0.8912	0.8452	0.8038	0.7662	0.7320	0.7007

Test results for the small $s = 7$ case is given in Table 1. It is clear that the experimental data is very close to the original theoretic values given by the iterative formula (11). This justifies our treatment of each DP sub-matrix $\mathtt{DM}_i$ as a normal DP matrix that contains a predictable number of duplicate starting points or chains. The main target of verification, which is the closed-form formula for $\frac{m_i}{m}$ given by Lemma 1, outputs numeric values that are slightly further away from the experimental data, but the error is by less than 2% at the worst.

The test results for more practical values of s are given in Figure 2. In order to present the larger data set for the larger s more effectively, we summarized the results as graphs. The lines represent our theory given by Lemma 1 and the

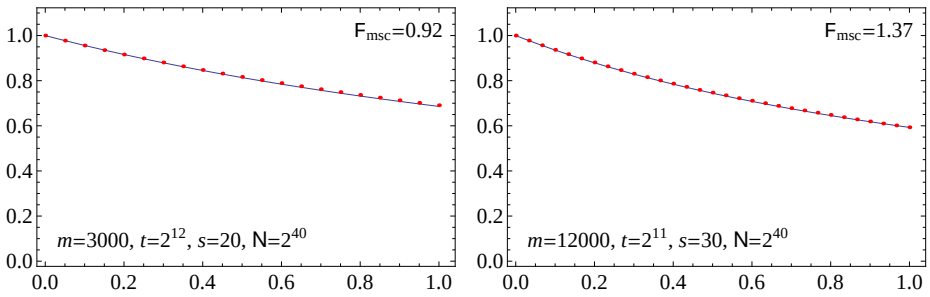

Fig. 2. Intermediate DP counts in a fuzzy rainbow matrix (*line*: theory; *dots*: test; x-axis: $\frac{i}{s}$; y-axis: $\frac{m_i}{m}$)

dots correspond to the experimental data. Each dot gives the count of the i-th color boundary points, averaged over ten tests. It is clear that the test results match our theory very well, despite the small number of test repetitions. We may conclude that Lemma 1 predicts the number of i-th color boundary points quite accurately for s values of interest and reasonably accurately for even very small values of s.

Our second experiment is designed to verify our arguments concerning the cost of resolving alarms. There are two main components to this argument. The first component is the probability for the online chains that start from the i-th color to be generated. This is tightly connected to the partial coverage rate formula (10), which has already been tested indirectly through our first experiment.

The second major component of the alarm cost argument is a technical claim made during the proof of Lemma 4. It concerns the probability of merge between a pre-computation chain and an online chain that starts from the i-th color. In more exact terms, we had stated the combined probability of merge at any one of the colors appearing strictly after the i-th color as $(s-i)\frac{t^2}{\mathsf{N}}$.

The experimental verification of this claim was carried out as follows. Multiple pre-computation chains were generated, and their color boundary DPs were recorded, rather than just their ending points. Since we were treating these chains as individual pre-computation chains, rather than as members of a pre-computation matrix, no sorting was done. After fixing a starting color index i, online chains were generated to the ending points. The first DP of the online chain, i.e., the DP that ends the i-th colored sub-chain, in addition to the terminating DP, were recorded. For each online chain, we did a linear search over the collection of pre-computation chains for matching ending points. Whenever a collision was found, we compared the first DP of the online chain against the corresponding color boundary DP of the colliding pre-computation chain to check whether the merge occur within the i-th color sub-chain or strictly after the i-th color, and occurrences of only the latter type of collisions were counted. Online chains that start at different color indices were tested for merges against a common large collection of pre-computation chains.

Since the matrix stopping rule plays no role in this experiment, we were free to use any large number of pre-computation chains, and test repetitions were not necessary. Only two sets of tests, corresponding to different parameter choices, were executed.

Table 2. Probability of merge between an online chain that starts from the i-th color and a pre-computation chain. Only merges at colors strictly after the i-th color are counted.

$t = 2^{12}$, $s = 20$, $\mathsf{N} = 2^{40}$, # of pre-computation chains = 500000							
i		1	4	8	12	16	19
# of online chains		26315	31250	41666	62500	125000	500000
# of merges	test	3753142	3763624	3795469	3841206	3847142	3728423
	theory	3814583	3814697	3814636	3814697	3814697	3814697
test/theory		0.9839	0.9866	0.9950	1.0069	1.0085	0.9774

$t = 2^{11}$, $s = 30$, $\mathsf{N} = 2^{40}$, # of pre-computation chains = 500000							
i		1	6	12	18	24	29
# of online chains		17241	20833	27777	41666	83333	500000
# of merges	test	966734	954600	948527	964926	979533	948478
	theory	953653	953659	953648	953659	953671	953674
test/theory		1.0137	1.0010	0.9946	1.0118	1.0271	0.9946

The experiment results are summarized in Table 2. The theoretic merge counts listed in the table were computed by multiplying the number of pre-computation chains and the number of online chains generated for the index i to the claimed probability $(s-i)\frac{t^2}{\mathsf{N}}$. The experimentally obtained merge counts for the two sets of tests are close to our theoretic predictions.

7 Comparison of Tradeoff Algorithms

Let us follow the framework of [19] in comparing the fuzzy rainbow tradeoff algorithm against the original rainbow tradeoff algorithm. That is, we present the pre-computation coefficient versus tradeoff coefficient curves for the non-perfect fuzzy rainbow, perfect rainbow, and non-perfect rainbow tradeoffs at several fixed success rate requirements. Since the tradeoff coefficient is a good measure of the resources required during the online phase, the curves present the range of options made available by each algorithm concerning the degree of online efficiency that can be achieved after the investment of certain amount of pre-computation effort. The perfect and non-perfect rainbow tradeoffs were chosen as the comparison targets, because the two were shown by the recent works [19, 20] to be the most competitive algorithms among the five major tradeoff algorithms, under typical conditions.

7.1 Drawing the curve

Let us explain how the $\mathtt{F}_{\mathrm{pc}}$ versus $\mathtt{F}_{\mathrm{tc},s}$ curve for the fuzzy rainbow tradeoff, under fixed $\mathtt{F}_{\mathrm{ps}}$ and s, may be drawn from the analysis results obtained so far in this work. The information required to draw the curves for the comparison target algorithms can be found in [19, 20]

Recall from Remark 1, given at the end of Section 3, that both the coverage rate $\mathtt{F}_{\mathrm{cr}}$ and the ratio $\frac{\ell}{t}$ may be seen as functions of the single variable $\mathtt{F}_{\mathrm{msc}}$, when parameters are restricted to those that achieve a fixed success rate. Hence, given any fixed s, we may view both the pre-computation coefficient

$$\mathtt{F}_{\mathrm{pc}} = \frac{\{-\ln(1-\mathtt{F}_{\mathrm{ps}})\}}{\mathtt{F}_{\mathrm{cr}}} \tag{16}$$

and the tradeoff coefficient $\mathtt{F}_{\mathrm{tc},s}$ of Theorem 1 as functions of the single parameter $\mathtt{F}_{\mathrm{msc}}$, when under a fixed success rate requirement $\mathtt{F}_{\mathrm{ps}}$. Given any $\mathtt{F}_{\mathrm{ps}}$ and s, the $\mathtt{F}_{\mathrm{pc}}$ versus $\mathtt{F}_{\mathrm{tc},s}$ curve can be drawn as a curve parameterized by $\mathtt{F}_{\mathrm{msc}}$.

It will become evident in the next subsection that the s values of interest are small. Hence, the summation appearing in the formula for $\mathtt{F}_{\mathrm{tc},s}$ does not bring about any practical difficulties in manipulations of the formulas, for example, drawing the curves or finding the lowest point on the curve.

7.2 Optimal s

Before comparing the fuzzy rainbow tradeoff with the original rainbow tradeoffs, let us first discuss the effects of the parameter s on the performance of the fuzzy rainbow tradeoff.

Consider a fuzzy rainbow tradeoff implementer that is working with a certain fixed s. Suppose that a set of parameters m, t, and ℓ achieving all desired properties have been found. These parameters need not be optimal in any particular sense, and we only assume that the desired success rate is met, while the balance between resource requirements, such as the pre-computation cost, physical storage size, and expected online time, is suitable for the intended purpose and environment. The implementer is willing to further tweak the parameters slightly, but large changes to $\log m$, $\log t$, and $\log \ell$, that would destroy the favorable balance between pre-computation cost, storage size, and online time will not be made.

The implementer still wishes to explore the possibility of using a different s values, as long as the three resource requirements remain largely unchanged. Consider another s-value s' and the associated parameter set $m' = \frac{s'}{s}m$, $t' = \frac{s}{s'}t$, and $\ell' = \frac{s}{s'}\ell$. It may be easier to read the material below with the simple example $m' = 2m$, $t' = \frac{t}{2}$, $s' = 2s$, and $\ell' = \frac{\ell}{2}$, in mind. It is clear that the pre-computation coefficient

$$\mathtt{F}_{\mathrm{pc}} = \frac{mts\ell}{\mathsf{N}} = \frac{m't's'\ell'}{\mathsf{N}} = \mathtt{F}'_{\mathrm{pc}} \tag{17}$$

and number of table entries

$$M = m\ell = m'\ell' = M', \tag{18}$$

for the new set of parameters, are identical to those of the original parameters. Furthermore,

$$\mathtt{F}_{\mathrm{msc}} = \frac{mt^2 s}{\mathsf{N}} = \frac{m't'^2 s'}{\mathsf{N}} = \mathtt{F}'_{\mathrm{msc}} \tag{19}$$

implies that the coverage rate $\mathtt{F}_{\mathrm{cr}}$ (Proposition 2) and success rate $\mathtt{F}_{\mathrm{ps}}$ (Proposition 1) also remain unchanged.

Since the algorithm behavior corresponding to the two sets of parameters seems to differ only in their time complexities T' and T, it may be tempting to simply compare $\mathtt{F}_{\mathrm{tc},s'} = \frac{T'M'^2}{\mathsf{N}^2} = \frac{T'M^2}{\mathsf{N}^2}$ against $\mathtt{F}_{\mathrm{tc},s} = \frac{TM^2}{\mathsf{N}}$ to determine which of the two are better. However, one must keep in mind that M does not represent the physical amount of storage. Recall from Section 5 that $\log m + \varepsilon$ bits of storage are required per fuzzy rainbow table entry, where ε corresponds to the bits used to record the partial ending points that remain after applications of the ending point truncation and the index table techniques. Hence, to be fair, one must compare

$$(\log m + \varepsilon)^2\, \mathtt{F}_{\mathrm{tc},s} \quad \text{and} \quad (\log m' + \varepsilon)^2\, \mathtt{F}_{\mathrm{tc},s'} = \left\{ \log m + \varepsilon + \log\left(\frac{s'}{s}\right) \right\}^2 \mathtt{F}_{\mathrm{tc},s'} \tag{20}$$

against each other. For example, when $s' = 2s$, i.e., if s is increased by one bit to s', one must compare $(\log m + \varepsilon)^2\, \mathtt{F}_{\mathrm{tc},s}$ against $(\log m + \varepsilon + 1)^2 \mathtt{F}_{\mathrm{tc},s'}$.

Let us briefly work with explicit example figures. The first box of Figure 3

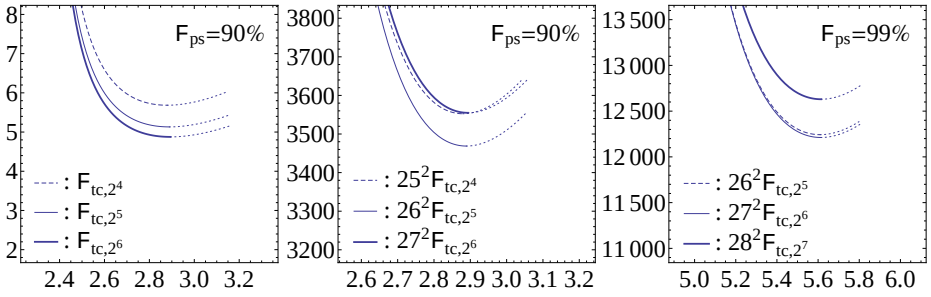

Fig. 3. The tradeoff coefficient $\mathtt{F}_{\mathrm{tc}}$ in relation to their respective pre-computation costs, at small s (x-axis: pre-computation coefficient; y-axis: tradeoff coefficients).

presents the $\mathtt{F}_{\mathrm{pc}}$ versus $\mathtt{F}_{\mathrm{tc},s}$ curves at 90% success rate, for the $s = 2^4$, $s = 2^5$, and $s = 2^6$ cases. The x-axis gives the pre-computation coefficient $\mathtt{F}_{\mathrm{pc}}$ and the y-axis gives the tradeoff coefficients $\mathtt{F}_{\mathrm{tc},s}$. The lower parts in each box correspond to better online efficiency and parts closer to the left edge correspond to smaller pre-computation requirements. The parameters corresponding to dotted parts of

the curves should not be used, since they correspond to worse online efficiency at more pre-computation cost than the lowest point of each curve. It is evident that increasing the s value brings the $\mathtt{F}_{\mathrm{pc}}$ versus $\mathtt{F}_{\mathrm{tc},s}$ curve closer to the bottom left corner. However, as previously noted, this observation alone should not be used to draw any premature conclusions.

Suppose that a certain parameter set appropriate for the resources available to the tradeoff implementer is such that $\log m + \varepsilon = 25$, when $s = 2^4$ is used. Under this situation, the discussion above shows that, in order to compare the performances of the fuzzy rainbow tradeoffs running with different s, we should be focusing on the second box of Figure 3 that present curves for the adjusted tradeoff coefficients. The change from $s = 2^4$ to $s = 2^5$ follows the trend seen in the first box, but the transition from $s = 2^5$ to $s = 2^6$ does not, worsening the performance of the fuzzy rainbow tradeoff. The choice of $s = 2^5$ is seen to be optimal, at least among the powers of 2, for this situation.

The optimal choice of s certainly would have been different if we had started from a different pairing of $\log m + \varepsilon$ and s. Comparison of the second and third boxes of Figure 3 shows that the optimal choice of s also depends on the success rate requirement.

Using a larger s has the positive effect of reducing the tradeoff coefficient $\mathtt{F}_{\mathrm{tc},s}$, but also has the negative effect of increasing the number of bits required to store each table entry. In order to find the value of s that is optimal for the specific situation in hand, it suffices to draw curves, similar to the second than third boxes of Figure 3, corresponding to various s options, with appropriate adjustment factors multiplied to $\mathtt{F}_{\mathrm{tc},s}$.

7.3 Fuzzy rainbow tradeoff versus rainbow tradeoff

To compare the fuzzy rainbow tradeoff directly with the perfect and non-perfect rainbow tradeoffs, we need to find the appropriate adjustment factors to be multiplied to the tradeoff coefficients, and this starts with a discussion of the fuzzy rainbow tradeoff parameters $m_{\mathtt{F}}$, $t_{\mathtt{F}}$, $\ell_{\mathtt{F}}$, and s that would make the resource requirements of the algorithm comparable to those of the perfect or non-perfect rainbow tradeoffs running under parameters $m_{\mathtt{R}}$, $t_{\mathtt{R}}$, and $\ell_{\mathtt{R}}$.

We had mentioned in Section 5 that $T_{\mathtt{F}} \approx t_{\mathtt{F}}^2 s^2$ and we know that $T_{\mathtt{R}} \approx t_{\mathtt{R}}^2$. Equating the very rough time and storage complexities of the two algorithms and using the facts $\ell_{\mathtt{F}} \approx t_{\mathtt{F}}$ (Remark 2) and $\ell_{\mathtt{R}} \approx 1$, we see that one must require

$$t_{\mathtt{F}}^2 s^2 \approx t_{\mathtt{R}}^2 \quad \text{and} \quad m_{\mathtt{F}} t_{\mathtt{F}} \approx m_{\mathtt{F}} \ell_{\mathtt{F}} \approx m_{\mathtt{R}} \ell_{\mathtt{R}} \approx m_{\mathtt{R}}. \tag{21}$$

These should be taken as extremely rough requirements, but the relations

$$\log t_{\mathtt{F}} + \log s \approx \log t_{\mathtt{R}} \quad \text{and} \quad \log m_{\mathtt{F}} + \log t_{\mathtt{F}} \approx \log m_{\mathtt{R}} \tag{22}$$

are somewhat reasonably accurate requirements one should adhere to, if the two algorithms are to be using similar resources.

Recall from [19] and [20] that each pre-computation table entry of both the perfect and non-perfect rainbow tradeoffs consumes $\log m_{\mathtt{R}} + \varepsilon_{\mathtt{R}}$ bits of storage,

where $\varepsilon_{\mathtt{R}}$ is a small positive integer. We have already seen that the fuzzy rainbow tradeoff similarly consumes $\log m_{\mathtt{F}} + \varepsilon_{\mathtt{F}}$ bits of storage per table entry.

Our interest lies in the ratio of bits required per table entry, and the use of (22) implies

$$\frac{\log m_{\mathtt{F}} + \varepsilon_{\mathtt{F}}}{\log m_{\mathtt{R}} + \varepsilon_{\mathtt{R}}} \approx \frac{\log m_{\mathtt{R}} - \log t_{\mathtt{R}} + \log s + \varepsilon_{\mathtt{F}}}{\log m_{\mathtt{R}} + \varepsilon_{\mathtt{R}}} \approx \frac{\frac{1}{3}\log \mathsf{N} + \log s + \varepsilon_{\mathtt{F}}}{\frac{2}{3}\log \mathsf{N} + \varepsilon_{\mathtt{R}}}, \tag{23}$$

where the second approximation is for the parameter set $m_{\mathtt{R}} = \mathsf{N}^{\frac{2}{3}}$ and $t_{\mathtt{R}} = \mathsf{N}^{\frac{1}{3}}$ that is typically considered during theoretic analyses of tradeoff algorithms. In the extreme, by ignoring the small integers $\varepsilon_{\mathtt{F}}$ and $\varepsilon_{\mathtt{R}}$, and also assuming s to be small, one might argue that this ratio could be as small as $\frac{1}{2}$, at the theoretically typical parameters. However, this is a rather optimistic figure that is biased in favor of the fuzzy rainbow tradeoff.

Consider the very large example of $\log \mathsf{N} \approx 75$ and the corresponding theoretically typical parameter set

$$\log m_{\mathtt{R}} \approx 50 \quad \text{and} \quad \log t_{\mathtt{R}} \approx 25, \tag{24}$$

for the rainbow tradeoff. According to (22), the parameter set for the fuzzy rainbow tradeoff of $s = 2^5$ that calls for comparable resources would satisfy

$$\log m_{\mathtt{F}} \approx 30 \quad \text{and} \quad \log t_{\mathtt{F}} \approx 20. \tag{25}$$

When $\varepsilon_{\mathtt{R}} \approx \varepsilon_{\mathtt{F}} \approx 7$, the ratio of bits per table entry is $\frac{\log m_{\mathtt{F}} + \varepsilon_{\mathtt{F}}}{\log m_{\mathtt{R}} + \varepsilon_{\mathtt{R}}} \approx \frac{37}{57}$. Hence, if one's favored balance of resources corresponds to rainbow tradeoff parameters of (24), one must compare the adjusted tradeoff coefficient $\left(\frac{37}{57}\right)^2 \mathtt{F}_{\mathrm{tc},2^5} \approx 0.42\,\mathtt{F}_{\mathrm{tc},2^5}$ against $\mathtt{R}_{\mathrm{tc}}$ (non-perfect rainbow) and $\bar{\mathtt{R}}_{\mathrm{tc}}$ (perfect rainbow).

Similarly, for the rather small example of $\log \mathsf{N} \approx 40$, $\log m_{\mathtt{R}} \approx 27$, $\log t_{\mathtt{R}} \approx 13$, $\varepsilon_{\mathtt{R}} \approx 7$, and $\varepsilon_{\mathtt{F}} \approx 7$, with $\log s \approx 5$, the ratio would be $\frac{\log m_{\mathtt{F}} + \varepsilon_{\mathtt{F}}}{\log m_{\mathtt{R}} + \varepsilon_{\mathtt{R}}} \approx \frac{13}{17}$, and a fair comparison would let $\left(\frac{13}{17}\right)^2 \mathtt{F}_{\mathrm{tc},2^5} = 0.58\,\mathtt{F}_{\mathrm{tc},2^5}$ compete against $\mathtt{R}_{\mathrm{tc}}$ and $\bar{\mathtt{R}}_{\mathrm{tc}}$.

Different $\mathtt{F}_{\mathrm{tc},s}$ adjustment factors will need to be applied, depending on s and the rough range of online resources that are appropriate for the situation in hand. However, taking the experience obtained from Figure 3 and the above two examples into account, we will somewhat arbitrarily choose to compare the two explicit choices $0.5\,\mathtt{F}_{\mathrm{tc},2^4}$ and $0.5\,\mathtt{F}_{\mathrm{tc},2^6}$ against $\mathtt{R}_{\mathrm{tc}}$ and $\bar{\mathtt{R}}_{\mathrm{tc}}$.

We clearly state that there could be situations that call for a tradeoff coefficient adjustment factor that is much larger than the 0.5 we will be using, in which case, the conclusions made below could be completely different. The appropriate adjustment factor depends not only on the externally given implementation environment, but also on the taste of the implementer concerning the balance between online efficiency and pre-computation cost, so that the adjustment factor cannot be fixed in an objective manner. In any case, the discussion given below can easily be adjusted to work for any specific situation.

The pre-computation coefficient versus tradeoff coefficient curves for the various tradeoff algorithms are given in Figure 4. Each box presents data corresponding to the success rate requirement indicated at its upper right corner.

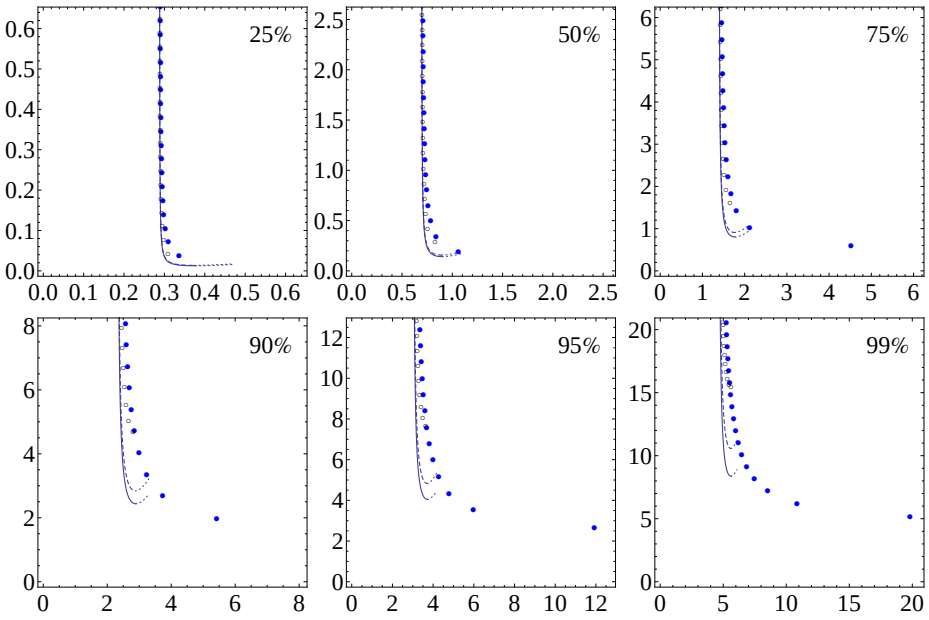

Fig. 4. The tradeoff coefficients $\mathtt{R}_{\mathtt{tc}}$ (empty circles), $\bar{\mathtt{R}}_{\mathtt{tc}}$ (filled dots), $0.5\,\mathtt{F}_{\mathtt{tc},2^4}$ (dashed), and $0.5\,\mathtt{F}_{\mathtt{tc},2^6}$ (line), in relation to their respective pre-computation costs, at various success rates (x-axis: pre-computation coefficients; y-axis: tradeoff coefficients).

The curves for the fuzzy rainbow tradeoffs are given as the dashed line ($s = 2^4$ case) and the thin line ($s = 2^6$ case). As mentioned before, the dotted parts of the curves should be ignored. The data for the perfect rainbow tradeoff (filled dots) and non-perfect rainbow tradeoff (empty circles) appear as discrete set of points, due to their small number of tables.

In all the boxes the two curves for the fuzzy rainbow tradeoff are situated closer to the lower left corner than the data points for the two rainbow tradeoffs. The fuzzy rainbow tradeoff provides better online efficiency for the same pre-computation investment. Thus a very rough conclusion would be that the fuzzy rainbow tradeoff is better in performance than the perfect and non-perfect rainbow tradeoffs.

At the 75% and higher success rates, the lowest dot for the perfect rainbow curve is situated lower than the lowest point of the two fuzzy rainbow tradeoff curves. This shows that the perfect rainbow tradeoff is able to provide better online efficiency than the fuzzy rainbow tradeoff at high success rates. That is, no choice of fuzzy rainbow tradeoff parameters will make its online efficiency better than the optimal efficiency reachable with the perfect rainbow tradeoff. Hence, the perfect rainbow tradeoff can be advantageous over the fuzzy rainbow tradeoff when the success rate requirement is high and the online efficiency is important.

However, it must be understood that the higher online efficiency option can be utilized only if it is paid for with higher pre-computation cost. Since the pre-computation cost is the largest barrier in any large scale deployment of the tradeoff technique, the higher cost cannot be ignored. At the high success rates, the decision as to whether the fuzzy rainbow or the perfect rainbow tradeoff is better will be different depending on how costly the additional pre-computation will be, relative to the value of better online efficiency, to the implementer.

At the low success rates 25% and 50%, the lowest point of the fuzzy rainbow curve is lower than the lowest point of the two original rainbow tradeoffs. For these low success rates, fuzzy rainbow tradeoff is always advantageous over the two original rainbow tradeoffs in terms of both the online efficiency and pre-computation cost.

Finally, a second pass through all six boxes, with focus on the empty circles, reveals that the performance of the non-perfect rainbow tradeoff is always inferior to that of the fuzzy rainbow tradeoff. The degree of online efficiency that can be provided by the non-perfect rainbow tradeoff can always be obtained with the fuzzy rainbow tradeoff at lower pre-computation cost.

An interesting observation that can be made from Figure 4 is that the distance between the two fuzzy rainbow tradeoff curves grows with the success rate requirement. Since the two curves were drawn with the same adjustment factor, this is an indication that the positive effect of increasing the s value on the tradeoff coefficient saturates much earlier at low success rates than at high success rates. Hence, small s values will be optimal at low success rates, and the possibility for larger s values to become optimal remains open at high success rates.

8 Conclusion

The online execution behavior of the non-perfect table fuzzy rainbow tradeoff was analyzed in this work. The success rate, the accurate average case online execution time that accounts for false alarms, and the physical storage size required to hold the pre-computation tables have all been obtained.

The information obtained through our analyses was used to compare the fuzzy rainbow tradeoff against the original rainbow tradeoff algorithm, which is widely believed to be the best tradeoff algorithm. The tradeoff coefficient adjustment factor and the color count s per table had to be fixed to somewhat arbitrary values for the comparison. However, our choices were based on figures obtained from reasonable examples, and the ensuing conclusions should be valid for the most part of the practical parameter range. Furthermore, the process can be repeated easily for any other choices, should there be the need to work with a drastically different range of parameters.

We discovered that the fuzzy rainbow tradeoff is always advantageous over the non-perfect rainbow tradeoff. The fuzzy rainbow tradeoff also outperforms the perfect rainbow tradeoff at low success rate requirements. For high success rate requirements, the situation is less conclusive. It is possible for the pre-

fect rainbow tradeoff to provide online efficiency that cannot be reached by the fuzzy rainbow tradeoff, but the advantage must be paid for with higher pre-computation cost. For efficiency levels that are reachable by both algorithms, the fuzzy rainbow tradeoff required less pre-computation.

It remains to analyze the perfect table version of the fuzzy rainbow tradeoff. The good performance of the non-perfect table fuzzy rainbow tradeoff witnessed through this work is an optimistic sign. On the other hand, the relatively poor performance of the perfect table DP tradeoff [20] could be interpreted as a negative indication.

References

1. Cryptohaze, GPU Rainbow Cracker. `https://www.cryptohaze.com/`
2. L0phtCrack, L0phtCrack 6. `http://www.l0phtcrack.com/`
3. Objectif Sécurité, Ophcrack. `http://ophcrack.sourceforge.net/`
4. RainbowCrack Project, RainbowCrack and RainbowCrack for GPU. `http://project-rainbowcrack.com/`
5. G. Avoine, P. Junod, P. Oechslin, Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Trans. Inform. Syst. Secur.*, **11**(4), 17:1–17:22 (2008). Preliminary version presented at INDOCRYPT 2005.
6. E. P. Barkan, Cryptanalysis of Ciphers and Protocols. Ph.D. Thesis, Technion—Israel Institute of Technology, March 2006.
7. E. Barkan, E. Biham, N. Keller, Instant ciphertext-only cryptanalysis of GSM encrypted communication. *Advances in Cryptology—CRYPTO 2003*, LNCS **2729**, (Springer, 2003), pp. 600–616.
8. E. Barkan, E. Biham, A. Shamir, Rigorous bounds on cryptanalytic time/memory tradeoffs. In *Advances in Cryptology—CRYPTO 2006*, LNCS **4117**, (Springer, 2006), pp. 1–21.
9. A. Biryukov, S. Mukhopadhyay, P. Sarkar, Improved time-memory trade-offs with multiple data. In *SAC 2005*, LNCS **3897**, (Springer-Verlag, 2006), pp. 110–127.
10. A. Biryukov, A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers, In *Advances in Cryptology—ASIACRYPT 2000*. LNCS **1976**, (Springer, 2000), pp. 1–13.
11. A. Biryukov, A. Shamir, D. Wagner, Real time cryptanalysis of A5/1 on a PC. In *FSE 2000*, LNCS **1978**, (Springer, 2001), pp. 1–18.
12. J. Borst, *Block Ciphers: Design, Analysis, and Side-Channel Analysis*. Ph.D. Thesis, Katholieke Universiteit Leuven, September 2001.
13. J. Borst, B. Preneel, J. Vandewalle, On the time-memory tradeoff betweeen exhaustive key search and table precomputation. In *Proceedings of the 19th Symposium on Information Theory in the Benelux*, WIC, 1998.
14. D. E. Denning, *Cryptography and Data Security*[1] (Addison-Wesley, 1982).
15. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. In *Advances in Cryptology—EUROCRYPT '97*, LNCS **1233**, (Springer, 1997), pp. 239–255.
16. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory*, **26**, (1980), pp. 401–406.

[1] On p.100, credit is given to Rivest for suggesting to apply the notion of distinguished points to the classical Hellman tradeoff.

17. Y. Z. Hoch, Security analysis of generic iterated hash functions. Ph.D. Thesis, Weizmann Institute of Science, August 2009.
18. J. Hong, G. W. Lee, D. Ma, Analysis of the parallel distinguished point tradeoff. In *Progress in Cryptogloy—INDOCRYPT 2011*, LNCS **7107**, (Springer, 2011), pp. 161–180.
19. J. Hong, S. Moon, A comparison of cryptanalytic tradeoff algorithms. To appear in *J. Cryptology*. Published online on July 2012 (http://dx.doi.org/10.1007/s00145-012-9128-3). Earlier version available from the Cryptology ePrint Archive as Report 2010/176.
20. G. W. Lee, J. Hong, A comparison of perfect table cryptanalytic tradeoff algorithms. Cryptology ePrint Archive, Report 2012/540.
21. K. Nohl, Attacking phone privacy. Presented at *Black Hat USA 2010*, Las Vegas, July 2010.
22. K. Nohl, C. Paget, GSM-SRSLY? Presented at *26th Chaos Communication Congress* (26C3), Berlin, December 2009.
23. P. Oechslin, Making a faster cryptanalytic time-memory trade-off. in *Advances in Cryptology–CRYPTO 2003*, LNCS **2729**, (Springer, 2003) pp. 617–630.
24. A. Shamir, Random Graphs in Security and Privacy. Invited talk at ICITS 2009.
25. F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, A time-memory tradeoff using distinguished points: New analysis & FPGA results. In *Cryptographic Hardware and Embedded Systems—CHES 2002*, LNCS **2523**, (Springer, 2003), pp. 593–609.