Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 765394, 19 pages
http://dx.doi.org/10.1155/2014/765394

*Research Article*

# Analysis of the Perfect Table Fuzzy Rainbow Tradeoff

**Byoung-Il Kim and Jin Hong**

*Department of Mathematical Sciences and ISaC, Seoul National University, Seoul 151-747, Republic of Korea*

Correspondence should be addressed to Jin Hong; jinhong@snu.ac.kr

Received 23 January 2014; Accepted 15 April 2014; Published 12 June 2014

Academic Editor: Igor Andrianov

Cryptanalytic time memory tradeoff algorithms are tools for inverting one-way functions, and they are used in practice to recover passwords that restrict access to digital documents. This work provides an accurate complexity analysis of the perfect table fuzzy rainbow tradeoff algorithm. Based on the analysis results, we show that the lesser known fuzzy rainbow tradeoff performs better than the original rainbow tradeoff, which is widely believed to be the best tradeoff algorithm. The fuzzy rainbow tradeoff can attain higher online efficiency than the rainbow tradeoff and do so at a lower precomputation cost.

## 1. Introduction

Cryptanalytic time memory tradeoff algorithms are tools for inverting generic one-way functions. They are actively used by law enforcement agencies and hackers to recover passwords protecting accesses to digital documents and to obtain system login passwords from the stored password hashes. After a one-time precomputation phase, whose computational complexity order is typical as that of an exhaustive computation of the one-way function on all inputs under consideration, a digest of the computation is written to a table of size that is of much smaller order than the complete dictionary. In the online phase, referencing the precomputation table, the input corresponding to a given inversion target is recovered with a computational complexity that is of much smaller order than that of an exhaustive trial of inputs. These algorithms allow tradeoffs to be made between the size of the precomputation table and the expected time for inversion through adjustments of various algorithm parameters.

The first time memory tradeoff method was the classical algorithm by Hellman [1] and this was soon followed by the distinguished points variant. Rivest is given credit [2, page 100] for suggesting to apply the notion of distinguished points to the classical Hellman tradeoff. Currently, the rainbow tradeoff [3] is the most widely used algorithm.

The fuzzy rainbow tradeoff [4, 5] is a more recent algorithm that combines the distinguished point and rainbow methods. The algorithm has already been used in the multitarget setting as an integral component of a fully functional attack [6, 7] on GSM phones, and an elementary analysis of the fuzzy rainbow tradeoff appeared in [8]. The latter work cites a work related to [6] and refers to the attack by the name Kraken, but none of these works cite the original publication [4, 5], indicating these to be an independent line of work. In fact, the analyses of [8] fall short of even the preliminary discussions given by [4, 5]. For now, the execution complexities of the fuzzy rainbow tradeoff are not known accurately enough for the purpose of comparing the performances of different tradeoff algorithms.

The fuzzy rainbow tradeoff, as with most other tradeoff algorithms, comes in the nonperfect table and perfect table versions. The perfect table version is expected to perform better during the online phase than the nonperfect version, but this must be paid for with a larger precomputation effort. Our previous work [9] gave an accurate performance analysis of the nonperfect table fuzzy rainbow tradeoff and compared the results with the performances of the original nonperfect and perfect table rainbow tradeoffs, which are widely believed to be the best tradeoff algorithms. The conclusions made there were that the nonperfect fuzzy rainbow tradeoff was always advantageous over the nonperfect rainbow tradeoff and that, while the perfect rainbow tradeoff could achieve somewhat better online efficiency than the nonperfect fuzzy rainbow tradeoff, for online efficiency levels that could be reached by both algorithms, the nonperfect fuzzy rainbow tradeoff could do so with a smaller amount of precomputation.

In this work, we analyze perfect table fuzzy rainbow tradeoff algorithm to present its performance accurately and compare this with the performances of the perfect rainbow tradeoff and the nonperfect fuzzy rainbow tradeoff. Our conclusion in rough terms is that the perfect fuzzy rainbow tradeoff outperforms the two comparison algorithms. This implies that the perfect fuzzy rainbow tradeoff, which has not yet received widespread recognition, is preferable to all the well-known tradeoff algorithms. We remark that the analysis given in this paper is completely different from that of our previous paper [9], which dealt with the nonperfect table case.

One clarification must be made concerning the subject algorithm of this work. The fuzzy rainbow tradeoff, as originally presented by [4, 5], was a tradeoff algorithm designed to be used in the multitarget setting. This is where the attacker is given multiple inversion targets and is deemed successful if he is able to recover the input corresponding to at least one of targets. However, our analysis of the algorithm in this paper will be done under the single-target setting.

Recall that a simple multitarget adaptation of the original rainbow tradeoff is quite inferior in performance [10] to the existing multitarget adaptations [11] of the classical Hellman and distinguished point tradeoffs and that the fuzzy rainbow tradeoff was designed to be a variant of the rainbow tradeoff that performs at a similar level. We have done some preliminary investigations and believe that it will not be difficult to transform the existing analysis results for the Hellman and distinguished point algorithms that were claimed for the single-target setting and the results of the present paper to the multitarget setting. In fact, we expect the existing equations concerning algorithm performances to remain essentially valid under the multitarget setting. Nevertheless, this transformation still requires a nontrivial amount of work and is relegated to a separate future work that focuses on the multitarget versions of the tradeoff algorithms. The current work will stay within the single-target setting.

The rest of this paper is organized as follows. In Section 2, we quickly review the fuzzy rainbow tradeoff algorithm and fix the notation. The execution behavior of the perfect table fuzzy rainbow tradeoff is fully analyzed in Section 3. This is a highly technical section and is the main contribution of this paper. Some experimental data that support the theoretical findings of this section are given in the appendix. In Section 4, we combine the results of our analysis with the existing analyses of other tradeoff algorithms to compare their performances. This could be more valuable to the practitioner than the details provided by Section 3. Finally, this paper is summarized in Section 5.

## 2. Preliminaries

Let us review the terminology concerning the fuzzy rainbow tradeoff algorithm and fix our notation. The reader is assumed to be familiar with the basic theory of the time memory tradeoff technique. In particular, we assume knowledge of the precomputation phase and online phase algorithms of the distinguished point (DP) and rainbow tradeoffs. The few sections in the beginning of [12] could be helpful in recalling these basics.

Throughout this paper, the one-way function $f : \mathcal{N} \to \mathcal{H}$ to be inverted is taken to act on a search space $\mathcal{N}$ of size $\mathsf{N}$. We fix $s$-many reduction functions $r_i : \mathcal{H} \to \mathcal{N}$ ($i = 1, \ldots, s$) and let $f_i : \mathcal{N} \to \mathcal{N}$ denote the composition of the one-way function $f$ and the reduction function $r_i$ of the $i$th color. The number of colors $s$ will typically be in the range $30 \leq s \leq 150$.

The structure of the precomputation matrix for the fuzzy rainbow tradeoff is a combination of those of the rainbow tradeoff and the DP tradeoff. One fixes a distinguishing property of probability $1/t$ and generates precomputation chains of the form

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

which could be referred to as a fuzzy rainbow chain. That is, one iterates the one-way function $f_i$ of a fixed color $i$ until the first appearance of a DP and the ending point of this DP subchain is used as the starting point for the next DP subchain. The color of the iteration function is changed at each intermediate DP until one reaches the end of the $s$th DP subchain. In short, each iteration of a rainbow chain is replaced by a DP chain, except that the number of colors used by each precomputation table is $s$ and that the expected chain length of each DP subchain is $t$.

Any implementation of an algorithm that relies on DPs to terminate a task must employ a mechanism to detect chains falling into loops. Typically, a bound on the chain length is set, and chains reaching this bound are discarded, possibly to be replaced with newly generated chains. We will assume the chain length bound is large enough to make the discarding of chains very infrequent, so that any effect the discarding may have on the algorithm performance can be ignored.

This paper deals with the perfect table version of the fuzzy rainbow tradeoff and the number of ending points for each perfect table is set to $m$. That is, one generates sufficiently many precomputation chains for each precomputation table, so that $m$ nonmerging precomputation chains can be collected. As with any tradeoff algorithm, the $m$ ordered pairs, each consisting of a starting point and an ending point, are sorted according to the ending points and recorded as the precomputation table. A total of $\ell$ precomputation tables are created during the precomputation phase.

The reader is cautioned to distinguish between the terms precomputation *table* and precomputation *matrix* while reading this paper. A precomputation table consists of just the starting and ending point pairs of the precomputation chains, whereas a matrix consists of all points of the precomputation chains, including the intermediate points that are not written to the table. The precomputation matrix is mentally visualized as a collection of chains, with some of them possibly merging into each other in the nonperfect case, rather than as a structureless set of points.

One can regard a single nonperfect fuzzy rainbow precomputation matrix as a concatenation of $s$-many nonperfect

DP submatrices. We will write $\mathtt{DM}_i$ to refer to the $i$th DP submatrix ($1 \leq i \leq s$) residing within a single nonperfect fuzzy rainbow matrix and use $|\mathtt{DM}_i|$ to denote the number of distinct points contained therein. The ending points of one DP submatrix become the starting points of the following DP submatrix, and the only difference between a standard nonperfect DP matrix and any $\mathtt{DM}_i$ is that the latter may contain duplicate starting points that lead to completely identical chains, should one insist on treating them as separate chains. The expected number of distinct starting points and ending points for $\mathtt{DM}_i$ is written as $m_{i-1}$ and $m_i$, respectively. In particular, $m_0$ is the number of starting points that are initially used in creating a perfect fuzzy rainbow matrix, and $m_s = m$ is the number of distinct terminal ending points of the fuzzy rainbow matrix.

The process of removing chain merges during the precomputation phase requires further clarification. The creation of a precomputation table for the perfect fuzzy rainbow tradeoff begins with a choice of $m_0$ starting points and the generation of the first nonperfect DP submatrix $\mathtt{DM}_1$. After the generation of each nonperfect DP submatrix $\mathtt{DM}_i$, the chains are sorted according to the ending points of $\mathtt{DM}_i$ and duplicate ending points are located to remove chain merges. Specifically, from each group of merging chains, one retains the chain with the longest $i$th color DP chain segment and discards the other chains. We denote the resulting (temporary) perfect DP submatrix as $\widetilde{\mathtt{DM}}_i$. The set of ending points from $\widetilde{\mathtt{DM}}_i$ is identical to the set of ending points from $\mathtt{DM}_i$, and these are used as the starting points for the next nonperfect DP submatrix $\mathtt{DM}_{i+1}$. For an appropriate choice of $m_0$, which will be discussed later, the final perfect DP submatrix $\widetilde{\mathtt{DM}}_s$ is expected to contain $m = m_s$ nonmerging chains. The collection of all DP chains in $\widetilde{\mathtt{DM}}_i$ that eventually reach one of the $m$ DP chains that remain in $\widetilde{\mathtt{DM}}_s$ is denoted by $\overline{\mathtt{DM}}_i$. In particular, we have $\overline{\mathtt{DM}}_s = \widetilde{\mathtt{DM}}_s$, and only the elements of the final perfect DP submatrices $\overline{\mathtt{DM}}_i$ can contribute to the success of inversions.

The method for handling merges explained above does not make reference to the total lengths of the chains and relies only on the $i$th DP chain segment lengths. We chose to work with such a merge removal rule, because it allowed existing results concerning the perfect DP tradeoff to be used during our analysis of the perfect fuzzy rainbow tradeoff. However, since some readers may object that it is more reasonable to base the merge removal rule on the total chain lengths, let us present two remarks concerning this matter.

First, we argue that the choice of merge removal method is not very important for the fuzzy rainbow tradeoff. Note that the rule for selecting one chain from a set of merging chains was an important issue for the perfect DP tradeoff that required attention because the chain lengths of a DP matrix form a geometric distribution. However, the lengths of the fuzzy rainbow chains form a distribution that very quickly approaches the normal distribution as $s$ is increased. Intuitively, this is to be expected, since the concatenation of multiple DP chains will create an averaging affect. In fact, it is not difficult to work out the distribution explicitly and verify the claim directly. Hence, the variation in fuzzy rainbow chain lengths is small, and the impact of choices based on chain lengths on the performance of the fuzzy rainbow tradeoff can only be limited. Furthermore, the averaging effect implies that, except at small $i$ values, our merge removal rule that references just the $i$th DP chain segment is likely to return the chain that is the longest in overall length.

Second, we question whether it is reasonable to retain the longer chains in the first place. The practice is widely accepted with the perfect DP tradeoff, because it is expected to bring about higher success rate for the same amount of storage use. However, the approach also increases both the number of false alarms and the average cost of resolving each false alarm. Although we strongly believe that the positive effect of choosing longer chains on the success rate is likely to outweigh its negative effects on the online cost, currently, there is no publicly available theoretical argument or experimental evidence to support such a claim. A separate detailed study would be required to arrive at a definitive answer concerning this matter.

This completes our description of the precomputation phase for the perfect fuzzy rainbow tradeoff. To the reader with some experience in the tradeoff technique, the online phase algorithm should now be mostly obvious from the structure of the precomputation matrix. Given an inversion target, for each precomputation table and starting color $1 \leq i \leq s$, one generates a partial fuzzy rainbow chain that starts from the $i$th color. If the terminal DP of this online chain can be found among the ending points of the precomputation table, the corresponding starting point is used to regenerate the precomputation chain, which could possibly return the correct input to the inversion target. However, most of the collisions will turn out to be false alarms, in which case the regeneration of the precomputation chain may be stopped at the DP for the color from which the online chain was started.

Some clarifications must be made concerning the order in which the online chains are created. In short, the multiple precomputation tables of the fuzzy rainbow tradeoff are processed in parallel, in a manner similar to the approach taken by the rainbow tradeoff. In practice, a round-robin style method can be used to simulate the parallel treatment of tables with even a single CPU, and this modification will not have a visible effect on the computational complexity, unless a small $s$ is used.

Let us make this more explicit. The online phase of the fuzzy rainbow tradeoff is performed in $s$ discrete steps. On the 1st step, the online DP chains for the $s$th colors, corresponding to the $\ell$ precomputation tables, are generated. All generated alarms are resolved before one moves onto the 2nd step. On the $i$th step, fuzzy rainbow chains that start from the $(s-i+1)$th colors, for the $\ell$ precomputation tables, are generated, and all resulting alarms are treated. The online phase is terminated when either the correct answer to the inversion target is found, or all $s$ steps have been completed. Even though it is likely for the answer to be obtained in the middle of the processing of some $i$th step, our analysis will assume that any step that has been initiated is fully completed, regardless of whether the answer has been secured. The effect of this simplification on the analysis results will be small, unless a small $s$ is used.

Our analysis of the perfect fuzzy rainbow tradeoff will frequently utilize two approximation techniques. The first is the approximation $(1-(1/b))^a \approx e^{-a/b}$. As explained in [12, Appendix A], this is appropriate when $a = O(b)$. The second technique is the approximation of a sum over a large index set into a definite integral. Both of these approximations will be very accurate, whenever we use them, as long as the tradeoff algorithm parameters are chosen reasonably. Throughout this paper, we will ignore multiplicative factors of $1 + O(1/t)$ size and write approximations of such order as equalities.

## 3. Analysis of the Perfect Table Fuzzy Rainbow Tradeoff

In this section, we analyze the online efficiency and the storage optimization issues for the perfect table fuzzy rainbow tradeoff. The expected computational complexity, rather than the worst case complexity, is computed and the effects of false alarms are fully taken into account. We always assume that the parameters $m$, $t$, and $s$, for the perfect fuzzy rainbow tradeoff, are chosen to satisfy $mt^2s = \overline{\mathtt{F}}_{\mathrm{msc}}\mathsf{N}$, with a matrix stopping constant $\overline{\mathtt{F}}_{\mathrm{msc}}$ that is neither too large nor very close to zero.

*3.1. Number of Color Boundary Points.* Let us consider a nonperfect fuzzy rainbow matrix created from $m_0$ starting points and its nonperfect DP submatrices $\mathtt{DM}_i$. For each $0 \le i \le s$, the collection of $m_i$ points that form the boundaries of the nonperfect DP submatrices will be referred to as the $i$th *color boundary points* of the nonperfect fuzzy rainbow matrix.

Our previous work [9] stated the relation

$$\left|\mathtt{DM}_i\right| = m_i t \tag{2}$$

and gave the iterative formula

$$\frac{m_i}{m_0} = \frac{m_{i-1}}{m_0}\,\frac{2}{1+\sqrt{1+2\left(\left(\mathtt{F}_{\mathrm{msc}}/s\right)\left(m_{i-1}/m_0\right)\right)}}, \tag{3}$$

for computing $m_i$, where $\mathtt{F}_{\mathrm{msc}} = (m_0t^2s)/\mathsf{N}$ is the matrix stopping constant for the nonperfect fuzzy rainbow matrix. The work also derived the closed-form approximation

$$m_i = \frac{2m_0}{2+\mathtt{F}_{\mathrm{msc}}\,(i/s)}, \tag{4}$$

under the assumption that $s$ is large, and claimed this to be accurate for even small $s$ values. The claimed accuracy is verified once more through experiments for parameters of our interest in the Appendix, and we will assume (4) is sufficiently accurate for the purpose of this work in the remainder of this paper.

Let us rewrite (4) in terms of the perfect fuzzy rainbow tradeoff parameters, so that the expression is more suitable for this work.

**Lemma 1.** *To create a perfect fuzzy rainbow matrix containing $m$ nonmerging chains, one must expect to generate $m_0 = (2/(2-\overline{\mathtt{F}}_{msc}))m$ chains. Furthermore, the number of $i$th color boundary points in the nonperfect fuzzy rainbow matrix generated during this process is expected to be*

$$m_i = \frac{2m}{\left(2-\overline{\mathtt{F}}_{msc}\right)+\overline{\mathtt{F}}_{msc}\,(i/s)}, \tag{5}$$

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

*Proof.* Substituting $i = s$ into (4), we know that a nonperfect fuzzy rainbow matrix created with $m_0$ starting points is expected to contain $m_s = (2/(2+\mathtt{F}_{\mathrm{msc}}))m_0$ nonmerging chains, where $\mathtt{F}_{\mathrm{msc}} = (m_0t^2s)/\mathsf{N}$. The requirement of $m = m_s = (2/(2+\mathtt{F}_{\mathrm{msc}}))m_0$ may be written as

$$\overline{\mathtt{F}}_{\mathrm{msc}} = \frac{2\mathtt{F}_{\mathrm{msc}}}{2+\mathtt{F}_{\mathrm{msc}}}. \tag{6}$$

Solving this equation for $\mathtt{F}_{\mathrm{msc}}$, we can rewrite it as

$$\mathtt{F}_{\mathrm{msc}} = \frac{2\overline{\mathtt{F}}_{\mathrm{msc}}}{2-\overline{\mathtt{F}}_{\mathrm{msc}}}, \tag{7}$$

which is equivalent to the first statement of this lemma. Substituting the first claim and the above equation into (4), we find

$$m_i = \frac{2\left(2/\left(2-\overline{\mathtt{F}}_{\mathrm{msc}}\right)\right)m}{2+\left(\left(2\overline{\mathtt{F}}_{\mathrm{msc}}/\left(2-\overline{\mathtt{F}}_{\mathrm{msc}}\right)\right)(i/s)\right)}, \tag{8}$$

and this is the second claim of this lemma. □

The first two displayed equations appearing in this proof both imply that $\overline{\mathtt{F}}_{\mathrm{msc}} < 2$ is always satisfied, which is similar to the situation with perfect rainbow tradeoffs. The reader may have guessed that taking $\overline{\mathtt{F}}_{\mathrm{msc}}$ very close to 2 corresponds to making bad parameter choices. In fact, we will later observe in Section 4.4 that $\overline{\mathtt{F}}_{\mathrm{msc}}$ is bounded sufficiently away from 2 for any meaningful parameters and that the precomputation requirement grows unrealistically large as $\overline{\mathtt{F}}_{\mathrm{msc}}$ approaches 2.

Because of its frequent appearances in the remainder of this section, we will introduce the notation

$$\overline{\mathtt{f}}_i = \frac{m_it^2}{\mathsf{N}} = \frac{2\overline{\mathtt{F}}_{\mathrm{msc}}}{\left(2-\overline{\mathtt{F}}_{\mathrm{msc}}\right)+\overline{\mathtt{F}}_{\mathrm{msc}}\,(i/s)}\,\frac{1}{s}. \tag{9}$$

When a small $s$ is in use, the symbol $\overline{\mathtt{f}}_i$ should be understood as designating the middle term, with the second equality interpreted as an approximation. However, we will mostly take the final term as the definition of $\overline{\mathtt{f}}_i$, assuming $s$ to be sufficiently large, and use this notation even for $i = s+1$ and $i = s+2$. Since $\overline{\mathtt{F}}_{\mathrm{msc}}$ is bounded away from 2 for all practical parameter sets, we may assume $\overline{\mathtt{f}}_i$ to be of $\Theta(1/s)$ order. Our use of the lower case letter $\overline{\mathtt{f}}$, rather than $\overline{\mathtt{F}}$, is meant to serve as a reminder that $\overline{\mathtt{f}}_i$ is not of $\Theta(1)$ order.

One can directly verify from the definition that

$$\frac{\overline{\mathtt{f}}_i}{\overline{\mathtt{f}}_j} = 1 + \frac{j-i}{2}\overline{\mathtt{f}}_i = \frac{1}{1+\left(\left((i-j)/2\right)\overline{\mathtt{f}}_j\right)}. \tag{10}$$

Combination of the middle expression with the knowledge of $\overline{\mathtt{f}}_i = \Theta(1/s)$ implies $(\overline{\mathtt{f}}_i/\overline{\mathtt{f}}_j) = \Theta(1)$, for $j \geq i$, and the right-hand side expression similarly implies the same claim, for $i \geq j$.

*3.2. Probability of Success.* An expression for the probability of success of the perfect fuzzy rainbow tradeoff is obtained in this subsection. We first define and present a formula for the *precomputation coefficient* of the algorithm.

**Proposition 2.** *The precomputation phase of the perfect table fuzzy rainbow tradeoff is expected to require* $\overline{\mathtt{F}}_{pc}\mathsf{N}$ *iterations of the one-way function, where the precomputation coefficient is*

$$\overline{\mathtt{F}}_{pc} = \frac{\ell}{t}\sum_{i=0}^{s-1}\frac{2\overline{\mathtt{F}}_{msc}}{(2-\overline{\mathtt{F}}_{msc}) + \overline{\mathtt{F}}_{msc}\,(i/s)}\frac{1}{s}. \tag{11}$$

*Proof.* Since each DP chain is expected to be of length $t$, on average, the computation of each temporary submatrix $\widetilde{\mathtt{DM}}_i$ from its $m_{i-1}$ distinct starting points requires $m_{i-1}t$ iterations of the one-way function. The effort of sorting the ending points of $\widetilde{\mathtt{DM}}_i$, so that duplicates can be removed and the distinct starting points for the next submatrix are obtained, is of $m \log m$ order, which is much smaller than the effort of generating the submatrix, and can be ignored. Taking account of the $\ell$ tables, the cost of precomputation can be stated as

$$(m_0 + m_1 + \cdots + m_{s-1})\, t\ell. \tag{12}$$

Applying Lemma 1, we can write this as

$$t\ell\sum_{i=0}^{s-1}\frac{2m}{(2-\overline{\mathtt{F}}_{\rm msc}) + \overline{\mathtt{F}}_{\rm msc}\,(i/s)} \tag{13}$$

to obtain the claimed formula. □

Let us use the notation

$$\overline{\mathtt{F}}_{{\rm cr},i} = \frac{|\overline{\mathtt{DM}}_i|}{mt}, \tag{14}$$

where $|\overline{\mathtt{DM}}_i|$ denotes the number of distinct points expected in the $i$th submatrix of a perfect fuzzy rainbow matrix, and define the *coverage rate* of a perfect fuzzy rainbow matrix to be

$$\overline{\mathtt{F}}_{\rm cr} = \frac{1}{mts}\sum_{i=1}^{s}|\overline{\mathtt{DM}}_i| = \frac{1}{s}\sum_{i=1}^{s}\overline{\mathtt{F}}_{{\rm cr},i}. \tag{15}$$

Note that the definitions allow us to expect both $\overline{\mathtt{F}}_{{\rm cr},i}$ and $\overline{\mathtt{F}}_{\rm cr}$ to be of $\Theta(1)$ order.

The coverage rate (15) of a perfect fuzzy rainbow matrix may be computed from $s$ and $\overline{\mathtt{F}}_{\rm msc}$ through the following formula.

**Lemma 3.** *The coverage rate of the DP submatrix* $\overline{\mathtt{DM}}_i$ *is given by*

$$\overline{\mathtt{F}}_{cr,i} = \frac{2\mathsf{N}}{m_i t^2}\ln\left(1 + \frac{m_i t^2}{2\mathsf{N}}\right) = \frac{2}{\overline{\mathtt{f}}_i}\ln\left(1 + \frac{\overline{\mathtt{f}}_i}{2}\right). \tag{16}$$

*Proof.* Recall that $\overline{\mathtt{DM}}_i$ is a subcollection of the chains appearing in $\widetilde{\mathtt{DM}}_i$. Note that the selection of chains from $\widetilde{\mathtt{DM}}_i$ to be retained in $\overline{\mathtt{DM}}_i$ depends on the behavior of the chains that extend out from the ending points of $\widetilde{\mathtt{DM}}_i$ and is independent of the $i$th submatrix itself. In other words, the chains of $\overline{\mathtt{DM}}_i$ have been selected at random from the chains of $\widetilde{\mathtt{DM}}_i$. Hence, the averages of chain lengths contained in $\widetilde{\mathtt{DM}}_i$ and $\overline{\mathtt{DM}}_i$ will be the same, and we can make the crucial observation that

$$\overline{\mathtt{F}}_{{\rm cr},i} = \frac{|\overline{\mathtt{DM}}_i|}{mt} = \frac{|\widetilde{\mathtt{DM}}_i|}{m_i t}. \tag{17}$$

In other words, the coverage rate of $\overline{\mathtt{DM}}_i$ is equal to the coverage rate of $\widetilde{\mathtt{DM}}_i$.

Now, recall that each $\widetilde{\mathtt{DM}}_i$ is simply a normal perfect DP matrix and also recall from [13] that the coverage rate of a perfect DP matrix $\overline{\mathtt{D}}_{\rm cr}$ may be computed as

$$\overline{\mathtt{D}}_{\rm cr} = \frac{2}{\overline{\mathtt{D}}_{\rm msc}}\ln\left(1 + \frac{\overline{\mathtt{D}}_{\rm msc}}{2}\right), \tag{18}$$

where $\overline{\mathtt{D}}_{\rm msc} = m' t^2/\mathsf{N}$ is the matrix stopping constant for the perfect DP matrix of $m'$ ending points. Thus we can write

$$\overline{\mathtt{F}}_{{\rm cr},i} = \frac{2}{m_i t^2/\mathsf{N}}\ln\left(1 + \frac{m_i t^2/\mathsf{N}}{2}\right), \tag{19}$$

as claimed. □

We are now ready to state the success probability of the perfect fuzzy rainbow tradeoff as a function of the algorithm parameters.

**Proposition 4.** *Consider an input to the one-way function that is chosen uniformly at random from the input space. Given the image of this input under the one-way function as the inversion target, the online phase of the perfect table fuzzy rainbow tradeoff will succeed in recovering the original input with probability*

$$\overline{\mathtt{F}}_{ps} = 1 - \exp\left(-\overline{\mathtt{F}}_{msc}\overline{\mathtt{F}}_{cr}\frac{\ell}{t}\right). \tag{20}$$

*Proof.* The probability of success one can expect from the online processing of a single DP submatrix $\overline{\mathtt{DM}}_i$ is $|\overline{\mathtt{DM}}_i|/\mathsf{N}$. Since the submatrices were generated by different reduction functions, we may treat them as being independent. Thus, the probability of success for the complete online phase, taking all the $\ell$ tables into account, may be written as

$$\overline{\mathtt{F}}_{\rm ps} = 1 - \prod_{i=1}^{s}\left(1 - \frac{|\overline{\mathtt{DM}}_i|}{\mathsf{N}}\right)^{\ell} = 1 - \prod_{i=1}^{s}\left(1 - \frac{\overline{\mathtt{F}}_{\rm msc}\overline{\mathtt{F}}_{{\rm cr},i}}{ts}\right)^{\ell}. \tag{21}$$

This can be approximated by

$$\overline{\mathtt{F}}_{\rm ps} = 1 - \exp\left(-\overline{\mathtt{F}}_{\rm msc}\sum_{i=1}^{s}\overline{\mathtt{F}}_{{\rm cr},i}\frac{\ell}{ts}\right) = 1 - \exp\left(-\overline{\mathtt{F}}_{\rm msc}\overline{\mathtt{F}}_{\rm cr}\frac{\ell}{t}\right), \tag{22}$$

as claimed. □

Given any set of parameters, one can compute the success rate by combining the above formula with definition (15), Lemma 3, and notation (9). More precisely, the success rate may be seen as a function of $\overline{\mathtt{F}}_{\mathrm{msc}}$, $\ell/t$, and $s$, rather than as a function of the more basic constant and parameters $\mathsf{N}$, $m$, $t$, $\ell$, and $s$.

The proposition also shows that one can fix positive integer $s$ and $\overline{\mathtt{F}}_{\mathrm{msc}} < 2$ to any value and still attain any success rate requirement $\overline{\mathtt{F}}_{\mathrm{ps}}$ by adhering to the relation

$$\frac{\ell}{t} = \frac{\left\{-\ln\left(1-\overline{\mathtt{F}}_{\mathrm{ps}}\right)\right\}}{\overline{\mathtt{F}}_{\mathrm{msc}}\overline{\mathtt{F}}_{\mathrm{cr}}}. \tag{23}$$

Note that this implies that, unless the requirement for the probability of success $\overline{\mathtt{F}}_{\mathrm{ps}}$ is unrealistically close to 1, we will have $\ell/t = \Theta(1)$. In other words, the parameters $\ell$ and $t$ will be of the same order.

*3.3. Online Complexity.* The time memory tradeoff curve for the perfect fuzzy rainbow tradeoff is obtained in this subsection through a careful computation of the average case online execution complexities. This is the most complicated part of this paper.

We start by assessing how likely each step of the online phase is to be executed.

**Lemma 5.** *The probability for an online chain that starts from the ith color of a perfect fuzzy rainbow matrix to be generated, that is, the probability for the ith DP submatrix $\overline{\mathtt{DM}}_i$ to be searched for the correct answer, is*

$$\exp\left(-\overline{\mathtt{F}}_{msc}\frac{\ell}{t}\frac{1}{s}\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k}\right) = \left(1-\overline{\mathtt{F}}_{ps}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k})/s\overline{\mathtt{F}}_{cr}}, \tag{24}$$

*where $\overline{\mathtt{F}}_{ps}$ is as given by Proposition 4.*

*Proof.* The online chain that starts from the $i$th color of a perfect fuzzy rainbow matrix will be generated if and only if the correct answer to the inversion target does not belong to the submatrices $\overline{\mathtt{DM}}_{i+1},\ldots,\overline{\mathtt{DM}}_s$ contained in the $\ell$ perfect fuzzy rainbow matrices. Hence, the probability under consideration is

$$\prod_{k=i+1}^{s}\left(1-\frac{\left|\overline{\mathtt{DM}}_k\right|}{\mathsf{N}}\right)^{\ell} = \exp\left(-\overline{\mathtt{F}}_{\mathrm{msc}}\frac{\ell}{t}\frac{1}{s}\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{\mathrm{cr},k}\right). \tag{25}$$

The equality claimed in the lemma statement follows from an application of Proposition 4. □

The cost of generating the online chains is a direct corollary to this lemma. It suffices to realize that an online chain that starts from the $i$th color is expected to be of length $t(s-i+1)$ and that there are $\ell$ tables to consider.

**Proposition 6.** *The generation of the online chains for the perfect fuzzy rainbow tradeoff is expected to require*

$$t\ell\sum_{i=1}^{s}(s-i+1)\left(1-\overline{\mathtt{F}}_{ps}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k})/s\overline{\mathtt{F}}_{cr}} \tag{26}$$

*iterations of the one-way function.*

Our next goal is to obtain the cost of resolving alarms that appear during the online phase. This part calls for very delicate arguments involving random functions and is very technical. The practice-oriented reader can skip the following few lemmas and jump to Proposition 10.

We will consider an online chain that starts from the $i$th color and treat the case of it merging into the fuzzy rainbow matrix within the $i$th color and the case of it merging at a strictly later color separately. As a preliminary result, we require the probability for an online chain to merge into a fuzzy rainbow matrix.

**Lemma 7.** *An online DP chain segment of the ith color will not merge into the nonperfect DP submatrix $\mathtt{DM}_i$ with probability $(1/(1+\overline{\mathtt{f}}_i)) = (\overline{\mathtt{f}}_{i+2}/\overline{\mathtt{f}}_i)$. The probability for an online chain that starts from the ith color not to merge into the perfect fuzzy rainbow precomputation matrix is $\prod_{k=i}^{s}(1/(1+\overline{\mathtt{f}}_k)) = (\overline{\mathtt{f}}_{s+1}/\overline{\mathtt{f}}_i)(\overline{\mathtt{f}}_{s+2}/\overline{\mathtt{f}}_{i+1})$.*

*Proof.* An online DP chain segment of the $i$th color will escape merging into the nonperfect DP submatrix $\mathtt{DM}_i$ with probability

$$\begin{aligned}\sum_{k=1}^{\infty}&\left(1-\frac{1}{t}-\frac{\left|\mathtt{DM}_i\right|}{\mathsf{N}}\right)^{k-1}\left(\frac{1}{t}-\frac{m_i}{\mathsf{N}}\right)\\ &=\frac{1-(tm_i/\mathsf{N})}{1+(t\left|\mathtt{DM}_i\right|/\mathsf{N})}\approx\frac{1}{1+(t\left|\mathtt{DM}_i\right|/\mathsf{N})},\end{aligned} \tag{27}$$

where the approximation is justified by the facts $(tm_i/\mathsf{N}) = \Theta(1/ts)$ and $(t|\mathtt{DM}_i|/\mathsf{N}) = \Theta(1/s)$. Substituting (2) into this expression and then applying (9), the probability can be written as

$$\frac{1}{1+\overline{\mathtt{f}}_i} = \frac{\left(2-\overline{\mathtt{F}}_{\mathrm{msc}}\right)+\overline{\mathtt{F}}_{\mathrm{msc}}\,(i/s)}{\left(2-\overline{\mathtt{F}}_{\mathrm{msc}}\right)+\overline{\mathtt{F}}_{\mathrm{msc}}\,((i+2)/s)} = \frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}. \tag{28}$$

The probabilities for an online chain to merge into $\overline{\mathtt{DM}}_k$ and $\mathtt{DM}_k$ are usually different, but the two are the same when $k = s$. Now, an online chain that starts from the $i$th color does not merge into the perfect precomputation matrix if and only if none of its DP chain segments merge into the submatrices $\overline{\mathtt{DM}}_i,\overline{\mathtt{DM}}_{i+1},\ldots,\overline{\mathtt{DM}}_s$, and this happens if and only if the online DP chain segments do not merge into the nonperfect submatrices $\mathtt{DM}_i,\mathtt{DM}_{i+1},\ldots,\mathtt{DM}_s$. Here, we emphasize that a series of submatrices ending at the terminal $s$th color is being considered. The claimed probability is the product of the term $(1/(1+\overline{\mathtt{f}}_k)) = ((\overline{\mathtt{f}}_{k+2})/\overline{\mathtt{f}}_k)$ over $k = i,\ldots,s$. □

The following lemma gives the cost of dealing with an alarm from a possible delayed merge, assuming the generation of an online chain that starts from the $i$th color.

**Lemma 8.** *Consider a single perfect fuzzy rainbow precomputation matrix and its associated s colors. Assume the generation of an online chain for this matrix that starts from the ith color. The cost of resolving an alarm that may be induced by* *a possible merge of this online chain into the fuzzy rainbow matrix strictly after the* $i$*th color is expected to be*

$$t\left(\sum_{k=1}^{i}\overline{\mathtt{F}}_{cr,k}\right)\left\{\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_{i+1}}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+2}}\right)+\left(1-\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\right)\left(1-\frac{\overline{\mathtt{f}}_{s}}{\overline{\mathtt{f}}_{i}}\right)\right\} \tag{29}$$

*iterations of the one-way function.*

*Proof.* We require the probability for an online chain that starts from the $i$th color to merge into the perfect fuzzy rainbow precomputation matrix without merging into the submatrix $\overline{\mathtt{DM}}_i$. Such a merge could occur through the following two separate events. (a) The $i$th color online DP chain segment does not merge into the nonperfect submatrix $\mathtt{DM}_i$, but the online chain part that extends out from the ending DP of the $i$th color segment merges into the perfect precomputation matrix. (b) The $i$th color online chain merges into $\mathtt{DM}_i$, without merging into $\overline{\mathtt{DM}}_i$, in which case the online chain is destined to merge into the perfect precomputation matrix at a later color.

If the $i$th color online DP chain segment does not merge into $\mathtt{DM}_i$, the chain that extends from its ending point may still be iterated with a random function. Hence, it is clear from Lemma 7 that the probability for the (a)-event to occur is

$$\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_{i+1}}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+2}}\right). \tag{30}$$

On the other hand, the extended part of the online chain is allowed no randomness in the (b)-event. We already know from the proof of Lemma 3 that an ending point of $\mathtt{DM}_i$, that is, an ending point of $\widetilde{\mathtt{DM}}_i$, has probability $(m_s/m_i) = (\overline{\mathtt{f}}_s/\overline{\mathtt{f}}_i)$ of remaining among the ending points of $\overline{\mathtt{DM}}_i$. Once again, Lemma 7 allows us to claim

$$\left(1-\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\right)\left(1-\frac{\overline{\mathtt{f}}_{s}}{\overline{\mathtt{f}}_{i}}\right) \tag{31}$$

as the probability for the (b)-event to occur. The probability for a merge to appear strictly after the $i$th color is the sum of the above two probabilities.

A merge of the online chain into the precomputation matrix that appears strictly after the $i$th color requires one to regenerate the associated precomputation chain up to the end of the $i$th color. Note that (14) allows us to write the average chain length $|\overline{\mathtt{DM}}_i|/m$ of each DP submatrix $\overline{\mathtt{DM}}_i$ as $\overline{\mathtt{F}}_{\mathrm{cr},i}t$. Hence, to resolve the alarm from the merge discussed above, one must expect to compute $(\overline{\mathtt{F}}_{\mathrm{cr},1}+\cdots+\overline{\mathtt{F}}_{\mathrm{cr},i})t$ iterations of the one-way function.

The claimed cost of resolving alarms is a simple product of the merge probability and the work factor we have already stated. □

The cost of dealing with an immediate merge of the online chain into the perfect precomputation matrix is given next.

**Lemma 9.** *Consider a single perfect fuzzy rainbow precomputation 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 an alarm that may be induced by a possible merge of this online chain into the fuzzy rainbow matrix within the* $i$*th color segment is expected to be approximately*

$$t\left(1+\sum_{k=1}^{i}\overline{\mathtt{F}}_{cr,k}\right)\left(1-\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\right)\frac{\overline{\mathtt{f}}_{s}}{\overline{\mathtt{f}}_{i}} \tag{32}$$

*iterations of the one-way function.*

*Proof.* Arguments given in the proof of Lemma 8 already show that the probability for a merge to occur within the $i$th color segment into the perfect DP submatrix is

$$\left(1-\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_{i}}\right)\frac{\overline{\mathtt{f}}_{s}}{\overline{\mathtt{f}}_{i}}. \tag{33}$$

Since the online chain will merge into at most one precomputation chain in $\overline{\mathtt{DM}}_i$, it only remains to find out how much work is required to resolve such a merge.

An alarm will require the associated precomputation chain to be regenerated at least up to the start of the $i$th DP submatrix, and we saw during the proof of Lemma 8 that this costs $(\overline{\mathtt{F}}_{\mathrm{cr},1}+\cdots+\overline{\mathtt{F}}_{\mathrm{cr},i-1})t$ iterations of the one-way function. The number of additional $i$th color segment iterations that are required must be handled more carefully. One can expect this to be larger than $\overline{\mathtt{F}}_{\mathrm{cr},i}t$, because longer precomputation chains are more likely to be involved in merges.

The work [12] stated in its Lemma 12 that the cost of resolving alarms expected from the processing of a single nonperfect DP table is $2\mathtt{D}_{\mathrm{msc}}t$ iterations of the one-way function, where $\mathtt{D}_{\mathrm{msc}}$ is the matrix stopping constant of the nonperfect DP matrix. Following the arguments given in the proof of the same lemma, one can write the number of false alarms expected during the processing of a single DP table as

$$\mathtt{D}_{\mathrm{msc}}\int_0^\infty\int_0^\infty \exp(-u)\exp(-v)\left\{\exp(\min(u,v))-1\right\}du\,dv = \mathtt{D}_{\mathrm{msc}}. \tag{34}$$

Computing the ratio of these two numbers, we can conclude that, during the processing of a single nonperfect DP table, each alarm calls for $2t$ iterations of the one-way function to resolve, on average.

Now, since (10) implies that $(m_i/m_{i-1}) = 1-\Theta(1/2s)$, we know that only a small portion of $\mathtt{DM}_i$ is discarded in creating $\widetilde{\mathtt{DM}}_i$, so that the DP matrices $\mathtt{DM}_i$ and $\widetilde{\mathtt{DM}}_i$ must be similar in their distributions of chain lengths. We also saw during the proof of Lemma 3 that the selection of $\overline{\mathtt{DM}}_i$ from $\widetilde{\mathtt{DM}}_i$ does not affect the distribution of chain lengths. Hence, it is reasonable to expect a merge of the online chain within the $i$th color segment into either $\mathtt{DM}_i$ or $\overline{\mathtt{DM}}_i$ to call for approximately $2t$ iterations of the $i$th colored one-way function, on average.

We wish to add a small tweak to the $2t$ claim. Note that the average chain length of a nonperfect DP matrix is $t$, whereas that of $\overline{\mathtt{DM}}_i$ is $\overline{\mathtt{F}}_{\mathrm{cr},i}t$. Since the average chain length may be understood as a concise representation of the distribution of chain lengths, one might expect chain lengths, and one might expect $2\overline{\mathtt{F}}_{\mathrm{cr},i}t$, which take the average chain length of $\overline{\mathtt{DM}}_i$ into account, to be a better approximation of the additional work than $2t$. In any case, since Lemma 3 implies that $\overline{\mathtt{F}}_{\mathrm{cr},i} = 1 - \Theta(1/4s)$, we know that $2t$ know that $2t$ and $2\overline{\mathtt{F}}_{\mathrm{cr},i}t$ are very close to each other. In similarity, we *choose* to take $(1 + \overline{\mathtt{F}}_{\mathrm{cr},i})t$ as the number of extra $i$th color iterations required, since this will make our later formulas look slightly simpler.

The claimed cost of resolving alarms is a simple combination of the three factors $(1 - (\overline{\mathtt{f}}_{i+2}/\overline{\mathtt{f}}_i))(\overline{\mathtt{f}}_s/\overline{\mathtt{f}}_i)$, $(\overline{\mathtt{F}}_{\mathrm{cr},1} + \cdots + \overline{\mathtt{F}}_{\mathrm{cr},i-1})t$, and $(1 + \overline{\mathtt{F}}_{\mathrm{cr},i})t$ that we have discussed so far. □

Combining Lemmas 8 and 9, we can state that the cost of dealing with an alarm that may be induced by an online chain generated from the $i$th color is

$$t\left\{\left(\sum_{k=1}^{i}\overline{\mathtt{F}}_{\mathrm{cr},k}\right)\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_i}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+1}}\right)+\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}\overline{\mathtt{f}}_s\right\}, \tag{35}$$

where we have relied on (10) to replace $(1 - (\overline{\mathtt{f}}_{i+2}/\overline{\mathtt{f}}_i))(\overline{\mathtt{f}}_s/\overline{\mathtt{f}}_i)$ with $(\overline{\mathtt{f}}_{i+2}/\overline{\mathtt{f}}_i)\overline{\mathtt{f}}_s$. Since, unlike other results of this paper, we have stated Lemma 9 only as an approximation, let us briefly explain that this is still very accurate. Using relation (10) and the rough approximations $\sum_{k=1}^{i}\overline{\mathtt{F}}_{\mathrm{cr},k} = \Theta(i)$ and $\overline{\mathtt{f}}_i = \Theta(1/s)$, it is easy to verify the facts

$$\begin{gathered}\left(\sum_{k=1}^{i}\overline{\mathtt{F}}_{\mathrm{cr},k}\right)\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_i}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+1}}\right)=\Theta\left(\frac{i\,(s-i+1)}{s}\right),\\ \frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}\overline{\mathtt{f}}_s=\Theta\left(\frac{1}{s}\right).\end{gathered} \tag{36}$$

This shows that, unless $s$ is very small, the first factor of (35) is at least $s$ times larger than the second factor. On the other hand, the proof of Lemma 9 shows that the error given by formula (35) will be more than sufficiently bounded by its second factor. Hence, we can expect (35) to be accurate up to a $1 + O(1/s)$ factor, at the worst. In fact, our testing to be presented in the Appendix confirms that the accuracy of (35) is much higher than what we have claimed here.

The computational cost of dealing with the alarms is now a direct corollary to Lemmas 5, 8, and 9.

**Proposition 10.** *The treatment of alarms during the online phase of a perfect fuzzy rainbow tradeoff is expected to require*

$$\begin{aligned}t\ell\sum_{i=1}^{s}&\left(1-\overline{\mathtt{F}}_{ps}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k})/s\overline{\mathtt{F}}_{cr}}\\ &\times\left\{\left(\sum_{k=1}^{i}\overline{\mathtt{F}}_{cr,k}\right)\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_i}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+1}}\right)+\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}\overline{\mathtt{f}}_s\right\}\end{aligned} \tag{37}$$

*iterations of the one-way function.*

The online computational complexity of the perfect fuzzy rainbow tradeoff can be stated as the sum

$$\begin{aligned}T = t\ell s\sum_{i=1}^{s}&\left(1-\overline{\mathtt{F}}_{\mathrm{ps}}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{\mathrm{cr},k})/s\overline{\mathtt{F}}_{\mathrm{cr}}}\\ &\times\left(\frac{s-i+1}{s}+\frac{1}{s}\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}\overline{\mathtt{f}}_s\right.\\ &\left.+\left(\frac{\sum_{k=1}^{i}\overline{\mathtt{F}}_{\mathrm{cr},k}}{s}\right)\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_i}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+1}}\right)\right)\end{aligned} \tag{38}$$

of its two components stated by Propositions 6 and 10. Recalling from the discussion given under (23) that $(\ell/t) = \Theta(1)$, it is easy to argue that $T$ is of $\Theta(t^2s^2)$ order. The following time memory tradeoff curve is a direct consequence of our knowledge secured of the online time complexity $T$ and the observation that the storage complexity is $M = m\ell$.

**Theorem 11.** *The time memory tradeoff curve for the perfect table fuzzy rainbow tradeoff is* $TM^2 = \overline{\mathtt{F}}_{tc}\mathsf{N}^2$, *where the tradeoff coefficient is*

$$\begin{aligned}\overline{\mathtt{F}}_{tc} = \overline{\mathtt{F}}_{msc}^{2}\left(\frac{\ell}{t}\right)^{3}\frac{1}{s}\sum_{i=1}^{s}&\left(1-\overline{\mathtt{F}}_{ps}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k})/s\overline{\mathtt{F}}_{cr}}\\ &\times\left(\frac{s-i+1}{s}+\frac{1}{s}\frac{\overline{\mathtt{f}}_{i+2}}{\overline{\mathtt{f}}_i}\overline{\mathtt{f}}_s\right.\\ &\left.+\left(\frac{\sum_{k=1}^{i}\overline{\mathtt{F}}_{cr,k}}{s}\right)\left(1-\frac{\overline{\mathtt{f}}_{s+1}}{\overline{\mathtt{f}}_i}\frac{\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i+1}}\right)\right).\end{aligned} \tag{39}$$

As a corollary to Lemma 5, we can state that the online phase of the perfect table fuzzy rainbow tradeoff is expected to call for

$$\ell\sum_{i=1}^{s}\left(1-\overline{\mathtt{F}}_{\mathrm{ps}}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{\mathrm{cr},k})/s\overline{\mathtt{F}}_{\mathrm{cr}}} \tag{40}$$

lookups to the precomputation table. One can easily verify that this is of $\Theta(ts)$ order, which is much smaller than the online computational complexity $T = \Theta(t^2s^2)$.

*3.4. Storage Optimization.* The storage complexity $M$ appearing in the tradeoff curve of Theorem 11 refers to the total number of entries that need to be stored in the precomputed tables. However, practical interest would be in the size of the required physical recording media expressed in number of bits. Hence, we need to discuss the number of bits occupied by each starting and ending point pair in the tables.

The naive approach would allocate $2\log\mathsf{N}$ bits to each table entry, but there are more efficient methods. Each starting point can be recorded in $\log m_0$ bits [14–16] through the use of consecutive starting points. In storing the ending points, one need not record bits that can be recovered from the definition of the distinguishing property [15, 17]. One can create an index table [15] for each sorted table and remove almost $\log m$ most significant bits from each ending point. Finally, one could simply truncate the ending points [5, 15] to a certain length before recording them, at the expense of dealing with additional false alarms arising from partial matches.

We will not explain the details of the methods mentioned above, since readable descriptions may be found in [12, 13]. Let us just provide the explicit relation between the degree of truncation and the amount of online computation increased by the associated false alarms.

**Proposition 12.** *Assume the use of the ending point truncation method in which the probability for two truncated randomly chosen DPs to be identical is* $1/r$. *Then, during the online phase of the perfect fuzzy rainbow tradeoff, one can expect to observe*

$$t\ell\frac{m}{r}\sum_{i=1}^{s}\left(1-\overline{\mathtt{F}}_{ps}\right)^{(\sum_{k=i+1}^{s}\overline{\mathtt{F}}_{cr,k})/s\overline{\mathtt{F}}_{cr}}\frac{\overline{\mathtt{f}}_{s+1}\overline{\mathtt{f}}_{s+2}}{\overline{\mathtt{f}}_{i}\overline{\mathtt{f}}_{i+1}}\sum_{k=1}^{i}\overline{\mathtt{F}}_{cr,k} \qquad (41)$$

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

*Proof.* The probability for the $\ell$ online chains that start from the $i$th color for each precomputation matrix to be generated is given by Lemma 5. The probability for such a generated chain not to merge into the perfect fuzzy rainbow precomputation matrix is given by Lemma 7. The probability for a nonmerging online chain to cause a truncation-related alarm with any one of the truncated ending points is $1/r$ and there are $m$ of these ending points, each of which could require separate treatment. Each alarm will require $(\overline{\mathtt{F}}_{\mathrm{cr},1}+\cdots+\overline{\mathtt{F}}_{\mathrm{cr},i})t$ iterations of the one-way function to resolve. Taking the $\ell$ precomputation matrices into account, the claimed formula is a simple combination of the facts mentioned so far. □

The cost stated above can easily be checked to be of $\Theta(t^2s^2(m/r))$ order. One can show, either by comparing this against $T = \Theta(t^2s^2)$ or through heuristic arguments, that the additional cost of resolving alarms induced by the ending point truncation technique can be suppressed to a negligible level by having the truncation retain slightly more than $\log m$ bits of information for each ending point. The arguments appearing in [12, 13] may then be repeated, almost word for word, to conclude that each entry of the perfect table fuzzy rainbow tradeoff can be recorded in $\log m_0 + \varepsilon$ bits, where $\varepsilon$ is a small positive integer.

The final conclusion made in the previous paragraph has not appeared before in the literature for the perfect fuzzy rainbow tradeoff. However, this is an expected result that has been obtained through straightforward adaptations of the arguments given in [12, 13].

## 4. Algorithm Comparison

This section may be slightly difficult to understand in full if the reader is unfamiliar with the approach that was recently introduced by [12] to compare different tradeoff algorithms and its further developments made by [13, 18]. However, we will refrain from providing lengthy repetitive explanation and justification of the comparison approach and ask the more interested reader to refer to the cited articles.

*4.1. Overview of the Comparison Method.* Recall that the conclusion of [13], in overly simplified terms, was that the perfect rainbow tradeoff algorithm is superior to all the other widely known tradeoff algorithms. Also recall from our recent work [9] that the nonperfect version of the fuzzy rainbow tradeoff, which is not yet widely known, is preferable to the perfect rainbow tradeoff when one is constrained in precomputation resources. Hence, we wish to compare the performance of the perfect fuzzy rainbow tradeoff, which we have analyzed in this paper, against those of the perfect rainbow and the nonperfect fuzzy rainbow tradeoffs.

In the remainder of this paper, we will use symbols $\overline{\mathtt{R}}$ and $\mathtt{F}$ to extend the notation we had been using concerning the perfect fuzzy rainbow tradeoff to the perfect rainbow and nonperfect fuzzy rainbow tradeoffs, and we will use the symbol $\mathtt{X}$ when we wish to reference a coefficient without making the tradeoff algorithm specific. For example, symbols $\overline{\mathtt{R}}_{\mathrm{tc}}$ and $\mathtt{F}_{\mathrm{tc}}$ denote the tradeoff coefficients for the perfect rainbow and nonperfect fuzzy rainbow tradeoffs, respectively, and $\mathtt{X}_{\mathrm{tc}}$ refers to any tradeoff coefficient. The symbols $\overline{\mathtt{F}}$, $\overline{\mathtt{R}}$, and $\mathtt{F}$ will also be used as in $m_{\overline{\mathtt{F}}}$, $m_{\overline{\mathtt{R}}}$, and $m_{\mathtt{F}}$ to clarify that the parameter or some complexity value is to be associated with a certain algorithm.

We will follow the approach of [12] in comparing the performances of different tradeoff algorithms against each other. In short, a small number of success rate requirements $\mathtt{X}_{\mathrm{ps}}$ will be chosen, and a graph for each algorithm, displaying the upper level tradeoff between its *precomputation cost* and *online cost*, will be plotted. The overall relative positions of the curves corresponding to different algorithms, subject to the same $\mathtt{X}_{\mathrm{ps}}$, will allow certain conclusions to be made concerning algorithm performances. The curves themselves will also be of value when choosing algorithm parameters for implementation.

It is clear that the precomputation cost $P$ of a tradeoff algorithm can be numerically represented in full by the precomputation coefficient $\mathtt{X}_{\mathrm{pc}} = (P/\mathsf{N})$. In the perfect fuzzy rainbow tradeoff case, this can be computed from Proposition 2, and the corresponding results for our two comparison target algorithms can be found in [9, 13]. Note that $\mathtt{X}_{\mathrm{pc}}$ presents the computational cost as the number of iterations, in multiples of $\mathsf{N}$, required of the common target one-way function, regardless of the tradeoff algorithm.

The online cost or efficiency of a tradeoff algorithm is mostly captured by its tradeoff coefficient. However, since the $M$ appearing in the definition $\mathtt{X}_{\mathrm{tc}} = TM^2/\mathsf{N}^2$ represents the number of table entries, rather than the physical number of bits required to store the tables, the tradeoff coefficient cannot be used directly in comparing different algorithms. One must first adjust $\mathtt{X}_{\mathrm{tc}}$ to account for the differences in number of bits required per table entry by the algorithms being compared. For example, the comparison of the nonperfect DP and rainbow tradeoffs by [12] was carried out with the *relatively* adjusted tradeoff coefficients $(1/4)\mathtt{D}_{\mathrm{tc}}$ and $\mathtt{R}_{\mathrm{tc}}$. The adjustment factor $(1/2)^2$ reflects the fact that the DP tradeoff requires roughly half as many bits to store each precomputation table entry in comparison to the rainbow tradeoff, under parameter choices that are typically considered during theoretical analyses of the tradeoff algorithms.

*4.2. Tradeoff Coefficient Adjustment.* We wish to be slightly more careful in treating the tradeoff coefficient adjustment factors than focusing on just the theoretically typical parameters. In general, as a trivial extension of the approach given by [12], one can use the *adjusted tradeoff coefficient*

$$\mathtt{X}_{\mathrm{atc}} = \left(\frac{3}{2\log \mathsf{N}}\right)^{2} \left(\begin{matrix}\text{number of bits per table}\\ \text{entry for } \mathtt{X}\end{matrix}\right)^{2} \mathtt{X}_{\mathrm{tc}} \tag{42}$$

to represent the online cost of each algorithm, and plot the $\mathtt{X}_{\mathrm{pc}}$ versus $\mathtt{X}_{\mathrm{atc}}$ curves to obtain a fair comparison of different algorithms. Here, the constant $(3/(2\log \mathsf{N}))^2$ serves the purpose of bringing the adjustment factor to $\Theta(1)$ order at typical parameters and is not an essential factor for the purpose of comparisons. The tradeoff coefficients $\overline{\mathtt{F}}_{\mathrm{tc}}$, $\overline{\mathtt{R}}_{\mathrm{tc}}$, and $\mathtt{F}_{\mathrm{tc}}$ may be computed from Theorem 11 and the corresponding results from [9, 13], but more work is required before we can specify the *number of bits* part more concretely for each algorithm.

Recalling the contents of Section 3.4 and Lemma 1, the adjusted tradeoff coefficient for the perfect fuzzy rainbow tradeoff may be written more concretely in terms of algorithm parameters as

$$\overline{\mathtt{F}}_{\mathrm{atc}} = \left(\frac{3}{2\log \mathsf{N}}\right)^{2} \left(\log \frac{2}{2-\overline{\mathtt{F}}_{\mathrm{msc}}} + \log m_{\overline{\mathtt{F}}} + \varepsilon\right)^{2} \overline{\mathtt{F}}_{\mathrm{tc}}. \tag{43}$$

Referring to results from [9, 13], it is not difficult to work out similar adjustment factors for the perfect rainbow and nonperfect fuzzy rainbow tradeoffs to claim

$$\begin{aligned}\overline{\mathtt{R}}_{\mathrm{atc}} &\\ &= \left(\frac{3}{2\log \mathsf{N}}\right)^{2} \left(\log \frac{2\ell_{\overline{\mathtt{R}}}}{2\ell_{\overline{\mathtt{R}}} + \ln\left(1-\overline{\mathtt{R}}_{\mathrm{ps}}\right)} + \log m_{\overline{\mathtt{R}}} + \varepsilon\right)^{2} \overline{\mathtt{R}}_{\mathrm{tc}},\end{aligned} \tag{44}$$

$$\mathtt{F}_{\mathrm{atc}} = \left(\frac{3}{2\log \mathsf{N}}\right)^{2} \left(\log m_{\mathtt{F}} + \varepsilon\right)^{2} \mathtt{F}_{\mathrm{tc}} \tag{45}$$

as the more concrete expressions.

The small positive integer $\varepsilon$ corresponds to the number of ending point bits remaining after applications of the truncation and index file techniques. Working with Proposition 12 and corresponding results from [13, 18], one can reasonably argue that $\varepsilon$ for the three algorithms can be set to a common value, which is why we have not subscripted them with the algorithm symbols. The terms involving $\overline{\mathtt{F}}_{\mathrm{msc}}$ and $\ell_{\overline{\mathtt{R}}}$, appearing in (43) and (44), have their roots in the merge removal process for producing perfect tables and reflect what fraction of the initially generated chains remains in the perfect table. Since the $\mathtt{X}_{\mathrm{pc}}$-$\mathtt{X}_{\mathrm{atc}}$ curves are to be plotted using $\overline{\mathtt{F}}_{\mathrm{msc}}$ and $\ell_{\overline{\mathtt{R}}}$ as parameters, the existence of these terms will not cause any later difficulties. However, the three $\log m$ values require further attention.

Note that one cannot hope to simply make independent $\log m$ value choices that achieve *optimality* for each algorithm, since optimality cannot be defined objectively. The most favorable balance between precomputation cost $P$, storage cost $M$, and expected online inversion time $T$ is a subjective matter that would be different for every implementer and situation. For our algorithm comparison to be fair, the values $\log m_{\overline{\mathtt{F}}}$, $\log m_{\overline{\mathtt{R}}}$, and $\log m_{\mathtt{F}}$ need to be left as choices to be made by the implementer. Nevertheless, we can still restrict the three choices to be made in a reasonably correlated manner. The natural approach is to correlate the choices through the requirement that the $P$, $M$, and $T$ complexities for the three algorithms be made roughly comparable and with the strict restriction that the success rates of the three algorithms be identical. Since the performances of the three algorithms commonly satisfy $TM^2 \approx \mathsf{N}^2$, if an implementer under a specific situation is asked to produce parameters sets for the three algorithms that he deems favorable, his choices for the three algorithms would be roughly matching the performance figures $P$, $M$, and $T$.

Let us now choose to view each positive integer $s$ as presenting a separate version of the perfect fuzzy rainbow algorithm. That is, we treat the perfect fuzzy rainbow tradeoff as a series of infinitely many different algorithms. A similar view will be taken of the nonperfect fuzzy rainbow algorithm. Below, we will describe a rule for correlating the parameter sets among these two infinite series of algorithms and the perfect rainbow tradeoff algorithm, for each fixed common probability of success $\mathtt{X}_{\mathrm{ps}}$.

We first take integers $m_\star$ and $t_\star$ such that $\log m_\star + 2\log t_\star \approx \log \mathsf{N}$ and set the perfect fuzzy rainbow parameters to

$$\begin{gathered} m_{\overline{\mathtt{F}},s} \approx s m_\star, \qquad t_{\overline{\mathtt{F}},s} \approx \frac{t_\star}{s}, \\ \ell_{\overline{\mathtt{F}},s} = -\frac{\ln\left(1-\mathtt{X}_{\mathrm{ps}}\right)}{\overline{\mathtt{F}}_{\mathrm{msc}}\overline{\mathtt{F}}_{\mathrm{cr}}}\, t_{\overline{\mathtt{F}},s}, \end{gathered} \tag{46}$$

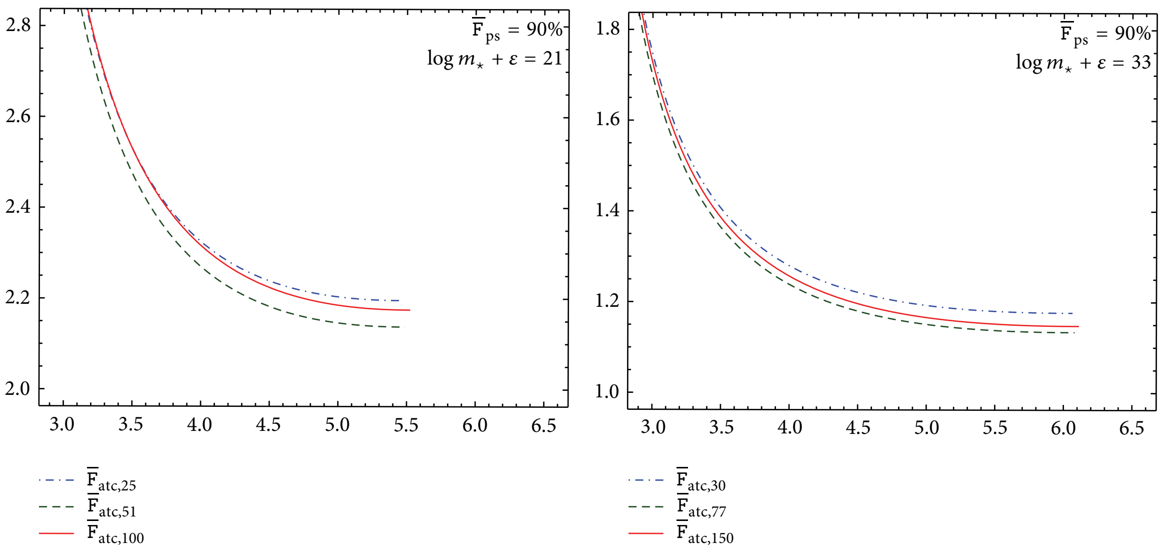
for each $s \geq 1$, where the coverage rate $\overline{\mathtt{F}}_{\mathrm{cr}}$, corresponding to each $s$, is to be computed through (9), Lemma 3, and (15). The perfect rainbow tradeoff parameters corresponding to the above are set to

$$m_{\overline{\mathtt{R}}} \approx m_\star t_\star, \qquad t_{\overline{\mathtt{R}}} \approx t_\star, \qquad \ell_{\overline{\mathtt{R}}} = -\frac{\ln\left(1-\mathtt{X}_{\mathrm{ps}}\right)}{\overline{\mathtt{R}}_{\mathrm{msc}}}, \tag{47}$$

where $\overline{\mathtt{R}}_{\mathrm{msc}} = (m_{\overline{\mathtt{R}}} t_{\overline{\mathtt{R}}}/\mathsf{N})$, and in such a way that $\ell_{\overline{\mathtt{R}}}$ is an integer. Finally, the nonperfect fuzzy rainbow parameters are set to

$$\begin{gathered} m_{\mathtt{F},s} \approx s m_\star, \qquad t_{\mathtt{F},s} \approx \frac{t_\star}{s}, \\ \ell_{\mathtt{F},s} = -\frac{\ln\left(1-\mathtt{X}_{\mathrm{ps}}\right)}{\mathtt{F}_{\mathrm{msc}}\mathtt{F}_{\mathrm{cr}}} t_{\mathtt{F},s}, \end{gathered} \tag{48}$$

for each $s \geq 1$, where $\mathtt{F}_{\mathrm{msc}} = (m_{\mathtt{F},s} t_{\mathtt{F},s}^2 s/\mathsf{N})$ and $\mathtt{F}_{\mathrm{cr}}$ is the coverage rate for the nonperfect fuzzy rainbow tradeoff.



(a)

(b)

Figure 1: The adjusted tradeoff coefficient $\overline{\mathtt{F}}_{\mathrm{atc},s}$ in relation to precomputation coefficient $\overline{\mathtt{F}}_{\mathrm{pc},s}$, for a small number of $s$ values, at success rate 90% and with $\log m_\star + \varepsilon$ fixed to 21 and 33 (bottom: $\overline{\mathtt{F}}_{\mathrm{pc},s}$; left: $\overline{\mathtt{F}}_{\mathrm{atc},s}$).

Note that the requirement $\log m_\star + 2\log t_\star \approx \log \mathsf{N}$ does not affect the implementer's control over the $P$-$T$-$M$ tradeoff in any way. Further note that one has sufficient control over $\overline{\mathtt{F}}_{\mathrm{msc}}$, $\ell_{\overline{\mathtt{R}}}$, and $\mathtt{F}_{\mathrm{msc}}$, even when $\log m_\star$ and $\log t_\star$ are fixed to approximate values. That is, one can vary the $\mathtt{X}_{\mathrm{pc}}$-$\mathtt{X}_{\mathrm{tc}}$ curve parameters $\overline{\mathtt{F}}_{\mathrm{msc}}$, $\ell_{\overline{\mathtt{R}}}$, and $\mathtt{F}_{\mathrm{msc}}$ quite freely while keeping all the $\log m_{\mathtt{X}}$ values somewhat stable.

Using (23) and similar claims from [9, 13], it is trivial to verify that all parameter sets achieve the same success rate $\mathtt{X}_{\mathrm{ps}}$. It is also easy to verify, using the facts $P_{\overline{\mathtt{F}}} = \Theta(m_{\overline{\mathtt{F}}} t_{\overline{\mathtt{F}}}^2 s_{\overline{\mathtt{F}}})$, $T_{\overline{\mathtt{F}}} = \Theta(t_{\overline{\mathtt{F}}}^2 s_{\overline{\mathtt{F}}}^2)$, and $M_{\overline{\mathtt{F}}} = \Theta(m_{\overline{\mathtt{F}}} t_{\overline{\mathtt{F}}})$ and the corresponding facts from [9, 13], that the performance figures $P$, $M$, and $T$ are of similar order for all the above parameter sets. In fact, it is possible to argue that the above method is the only possible manner in which comparable $P$, $T$, and $M$ complexities can be achieved by the different algorithms at a common success rate.

The reasonable association between the parameter sets for different algorithms given by (46), (47), and (48) allows the adjusted tradeoff coefficients to be written as

$$\overline{\mathtt{F}}_{\mathrm{atc},s} = \left(\frac{3}{2\log \mathsf{N}}\right)^2 \left(\log \frac{2}{2-\overline{\mathtt{F}}_{\mathrm{msc}}} + \log s + \log m_\star + \varepsilon\right)^2 \overline{\mathtt{F}}_{\mathrm{tc},s}, \tag{49}$$

$$\overline{\mathtt{R}}_{\mathrm{atc}} = \left(\frac{3}{2\log \mathsf{N}}\right)^2 \times \left(\log \frac{2\ell_{\overline{\mathtt{R}}}}{2\ell_{\overline{\mathtt{R}}} + \ln\left(1-\overline{\mathtt{R}}_{\mathrm{ps}}\right)} + \frac{1}{2}\log \mathsf{N} + \frac{1}{2}\log m_\star + \varepsilon\right)^2 \overline{\mathtt{R}}_{\mathrm{tc}}, \tag{50}$$

$$\mathtt{F}_{\mathrm{atc},s} = \left(\frac{3}{2\log \mathsf{N}}\right)^2 \left(\log s + \log m_\star + \varepsilon\right)^2 \mathtt{F}_{\mathrm{tc},s}, \tag{51}$$

where we have relied on $\log \mathsf{N} \approx \log m_\star + 2\log t_\star$ in writing (50). We have also additionally subscripted the (adjusted) tradeoff coefficients with the parameter $s$ to make their dependence on $s$ more explicit.

To compare the performances of different tradeoff algorithms, it now suffices to fix $\mathtt{X}_{\mathrm{ps}}$, choose reasonable values for $\log m_\star + \varepsilon$ and $\log \mathsf{N}$, and plot the $\mathtt{X}_{\mathrm{pc}}$-$\mathtt{X}_{\mathrm{atc}}$ curves, for all the algorithms.

*4.3. Choosing the Color Count s.* Before comparing the perfect fuzzy rainbow tradeoff with the other two algorithms, we wish to make comparisons among the many versions of the perfect fuzzy rainbow tradeoff corresponding to different $s$ choices. A small number of the $\overline{\mathtt{F}}_{\mathrm{pc}}$-$\overline{\mathtt{F}}_{\mathrm{atc}}$ curves are given in Figure 1.

The left-hand side box contains plots of the $(\overline{\mathtt{F}}_{\mathrm{pc},s}, \overline{\mathtt{F}}_{\mathrm{tc},s})$ points, for the case when the success rate is set to 90% and $\log m_\star + \varepsilon$ is set to 21. Each of the three curves is for specific $s$ values. The right-hand side box contains similar curves for the $\overline{\mathtt{F}}_{\mathrm{ps}} = 90\%$ and $\log m_\star + \varepsilon = 33$ case. Note that $\varepsilon = 8$ would be a reasonable value in view of the discussions given by Section 3.4 and that $\log m_\star = 13$ and $\log m_\star = 25$ are choices that would typically be made during theoretical discussions of the tradeoff technique for the very small $\mathsf{N} = 2^{39}$ and the very large $\mathsf{N} = 2^{75}$ search space sizes. Hence, our $\log m_\star + \varepsilon = 21$ and $\log m_\star + \varepsilon = 33$ examples are representative of the behaviors at both ends of the realistic tradeoff application environments.

Each curve has been plotted precisely to its lowest point. The curve points that would appear to the right of a lowest point are meaningless, since they correspond to parameter sets that call for higher precomputation efforts while achieving lower online efficiencies than the parameter sets corresponding to the lowest point.

One may roughly associate overall better performance with a curve that is closer to the lower left corner of each figure box. However, it can be seen that curves corresponding to different $s$ values may cross over each other. In fact, we could verify that the lowest curve in each box crosses over the middle curve appearing in its box at a high $\mathtt{X}_{\text{atc}}$ value. Hence, one cannot claim one $s$ value to be providing definitely superior performance over another $s$ value, at least not strictly logically.

Nevertheless, if we restrict our attention to the lower parts of the curves, which are of higher practical interest than the high $\mathtt{X}_{\text{atc}}$ parts, it becomes clear that little harm will be done by simplifying matters and ranking the performances for different $s$ choices based only on the lowest point of each curve. In other words, for practical purposes, it is sufficiently reasonable to declare an $s$ value to be optimal for a specific $(\overline{\mathtt{F}}_{\text{ps}}, \log m_\star + \varepsilon)$ pair if

$$\min_{\overline{\mathtt{F}}_{\text{msc}}} \left\{\overline{\mathtt{F}}_{\text{atc},s}\right\} = \min_{i \geq 1, \overline{\mathtt{F}}_{\text{msc}}} \left\{\overline{\mathtt{F}}_{\text{atc},i}\right\}. \tag{52}$$

We emphasize that such a simplification is possible only because of the nice relative positions of the many curves and that the same simplification may not be applicable to other comparison situations.

The optimal number of colors $s$, as defined through comparisons at the lowest points of the curves, is given by Table 1, for various success rate requirements $\overline{\mathtt{F}}_{\text{ps}}$ and a wide range of $\log m_\star + \varepsilon$ values. The $\overline{\mathtt{F}}_{\text{msc}}$ value that attains the minimum possible $\overline{\mathtt{F}}_{\text{atc},s}$ value is listed under each optimal $s$.

*4.4. Some Observation.* It can be seen from Table 1 that the optimal $s$ value becomes larger as the success rate requirement is increased and also as the number of bits per table entry becomes larger. The same trend could be observed from an analogous table presented by [18] for the nonperfect fuzzy rainbow tradeoff.

An intuitive explanation for the trend concerning the success rate can be given as follows. Since the online phase of the fuzzy rainbow tradeoff processes the precomputations tables in parallel, a higher $s$ value, corresponding to higher segmentation of the precomputation matrix, allows for more immediate exit from the online phase upon an encounter with the correct answer. Early exits from the online phase are less common under low success rates, and the importance of higher $s$ value increases as the success rate requirement is increased. As for the trend related to the change in $\log m_\star + \varepsilon$, one could treat this as a natural scaling effect that accompanies the general increase in search space size associated with the increase in $\log m_\star$.

Let us now return to the definition of the adjusted tradeoff coefficient. A careful reading of [13] shows that they had ignored the term involving $\ell_{\overline{\mathtt{R}}}$ from (50) in comparing the perfect rainbow tradeoff with the other major tradeoff algorithms. This was justified in their work based on the observation that the term remains upper bounded by a small number, when parameters are restricted to those that do not call for impractically large precomputation efforts.

Applying the same argument to the perfect fuzzy rainbow tradeoff, we can see that the $\log(2/(2-\overline{\mathtt{F}}_{\text{msc}}))$ term of (49) is likewise upper bounded for parameters that are reasonable in view of precomputation cost. Hence, we could consider the possibility of using the adjusted tradeoff coefficient defined as

$$\overline{\mathtt{F}}'_{\text{atc},s} = \left(\frac{3}{2\log \mathsf{N}}\right)^2 \left(\log s + \log m_\star + \varepsilon\right)^2. \tag{53}$$

This definition could be favorable to the previous definition (49), in view of simplicity.

The effect of removing the $\log(2/(2-\overline{\mathtt{F}}_{\text{msc}}))$ term can be seen from Figure 2. The curves for $\overline{\mathtt{F}}_{\text{atc},s}$ and $\overline{\mathtt{F}}'_{\text{atc},s}$ are noticeably different from each other, and the use of the simpler (53) in place of the more accurate (49) cannot be justified.

An overview of Table 1 reveals that any reasonable choice of parameters will mostly satisfy the bound $\overline{\mathtt{F}}_{\text{msc}} \leq 1.8$, and this bound implies that the term $\log(2/(2-\overline{\mathtt{F}}_{\text{msc}})) \leq 3.32193$ is always somewhat small. However, referring once more to Table 1, we see that $\log s$ values of interest are not very large. Furthermore, practical values of $\log m_\star + \varepsilon$ are not very large either. Hence, unlike the situation with the perfect rainbow tradeoff, the bound on $\log(2/(2-\overline{\mathtt{F}}_{\text{msc}}))$ is not small enough, in comparison to the terms it is added together with, to be completely ignored.

Our final comment is intimately connected to the above discussion. Combining the bound $\overline{\mathtt{F}}_{\text{msc}} \leq 1.8$ with Proposition 2 and (23), we can state the bound

$$\overline{\mathtt{F}}_{\text{pc}} \leq -\frac{\ln\left(1-\overline{\mathtt{F}}_{\text{ps}}\right)}{\overline{\mathtt{F}}_{\text{msc}}\overline{\mathtt{F}}_{\text{cr}}} \frac{2\overline{\mathtt{F}}_{\text{msc}}}{2-\overline{\mathtt{F}}_{\text{msc}}} \leq -\frac{\ln\left(1-\overline{\mathtt{F}}_{\text{ps}}\right)}{\Theta(1)}\, 10, \tag{54}$$

concerning the precomputation coefficient. Hence, unless the success rate requirement is set unrealistically close to 1, precomputation cost will be automatically bounded by $\Theta(\mathsf{N})$ for all meaningful parameters.

*4.5. Algorithm Comparison.* We are finally ready to compare the performance of the perfect fuzzy rainbow tradeoff with those of the perfect rainbow tradeoff and the nonperfect fuzzy rainbow tradeoff.

The comparison at the 90% success rate is given by Figure 3, which presents the $\mathtt{X}_{\text{pc}}$-$\mathtt{X}_{\text{atc}}$ curves for the three tradeoff algorithms. The left-hand side box was drawn with parameters that would be used with a very small search space and the right-hand side box represents the situation of a very large search space. The $s$ values for the perfect fuzzy rainbow tradeoffs were chosen to be the optimal ones given by Table 1. The corresponding table from [18] was used to decide on the $s$ values for the nonperfect fuzzy rainbow tradeoff.

The comparisons at 99% and 99.9% success rates are given by Figures 4 and 5. As before, parameters typically considered during theoretical analyses of the tradeoff algorithms corresponding to a small search space and a large search space were used.

Table 1: Optimal $s$ for the perfect fuzzy rainbow tradeoff, at various $\overline{\mathrm{F}}_{\mathrm{ps}}$ and $\log m_\star + \varepsilon$ values. The $\overline{\mathrm{F}}_{\mathrm{msc}}$ value listed below each $s$ attains the minimum $\overline{\mathrm{F}}_{\mathrm{atc},s}$ value.

| $\log m_\star + \varepsilon$ | | 50% | 75% | 90% | 95% | 99% | 99.5% | 99.9% |
|---|---|---|---|---|---|---|---|---|
| 18 | $s$ | 34 | 38 | 43 | 48 | 60 | 66 | 79 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.6880 | 1.6882 | 1.6846 | 1.6813 | 1.6697 | 1.6647 | 1.6531 |
| 19 | $s$ | 36 | 40 | 46 | 50 | 63 | 68 | 83 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.6999 | 1.6996 | 1.6967 | 1.6921 | 1.6810 | 1.6754 | 1.6644 |
| 20 | $s$ | 37 | 42 | 48 | 53 | 65 | 71 | 86 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7095 | 1.7103 | 1.7071 | 1.7030 | 1.6910 | 1.6858 | 1.6747 |
| 21 | $s$ | 39 | 44 | 50 | 55 | 68 | 74 | 89 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7198 | 1.7202 | 1.7167 | 1.7126 | 1.7008 | 1.6955 | 1.6843 |
| 22 | $s$ | 41 | 46 | 52 | 57 | 71 | 77 | 93 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7294 | 1.7293 | 1.7256 | 1.7215 | 1.7101 | 1.7046 | 1.6936 |
| 23 | $s$ | 43 | 48 | 54 | 59 | 73 | 80 | 96 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7382 | 1.7379 | 1.7339 | 1.7298 | 1.7183 | 1.7133 | 1.7021 |
| 24 | $s$ | 45 | 50 | 56 | 62 | 76 | 83 | 100 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7465 | 1.7459 | 1.7418 | 1.7381 | 1.7264 | 1.7214 | 1.7105 |
| 25 | $s$ | 47 | 51 | 58 | 64 | 79 | 86 | 103 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7542 | 1.7527 | 1.7492 | 1.7454 | 1.7340 | 1.7290 | 1.7180 |
| 26 | $s$ | 49 | 53 | 60 | 66 | 81 | 89 | 106 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7615 | 1.7598 | 1.7562 | 1.7523 | 1.7410 | 1.7362 | 1.7252 |
| 27 | $s$ | 51 | 55 | 63 | 69 | 84 | 91 | 110 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7684 | 1.7665 | 1.7633 | 1.7593 | 1.7478 | 1.7427 | 1.7322 |
| 28 | $s$ | 52 | 57 | 65 | 71 | 87 | 94 | 113 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7740 | 1.7728 | 1.7695 | 1.7655 | 1.7543 | 1.7492 | 1.7387 |
| 29 | $s$ | 54 | 59 | 67 | 73 | 89 | 97 | 116 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7801 | 1.7788 | 1.7754 | 1.7713 | 1.7602 | 1.7554 | 1.7448 |
| 30 | $s$ | 56 | 61 | 69 | 75 | 92 | 100 | 120 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7859 | 1.7845 | 1.7809 | 1.7769 | 1.7661 | 1.7612 | 1.7508 |
| 31 | $s$ | 58 | 63 | 71 | 78 | 95 | 103 | 123 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7914 | 1.7898 | 1.7862 | 1.7825 | 1.7716 | 1.7668 | 1.7564 |
| 32 | $s$ | 60 | 65 | 73 | 80 | 97 | 106 | 126 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.7966 | 1.7949 | 1.7913 | 1.7875 | 1.7767 | 1.7720 | 1.7618 |
| 33 | $s$ | 62 | 67 | 75 | 82 | 100 | 109 | 130 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.8015 | 1.7998 | 1.7961 | 1.7923 | 1.7817 | 1.7771 | 1.7670 |
| 34 | $s$ | 64 | 69 | 78 | 84 | 103 | 111 | 133 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.8062 | 1.8044 | 1.8010 | 1.7969 | 1.7865 | 1.7818 | 1.7718 |
| 35 | $s$ | 66 | 71 | 80 | 87 | 105 | 114 | 136 |
| | $\overline{\mathrm{F}}_{\mathrm{msc}}$ | 1.8106 | 1.8088 | 1.8053 | 1.8015 | 1.7909 | 1.7864 | 1.7765 |

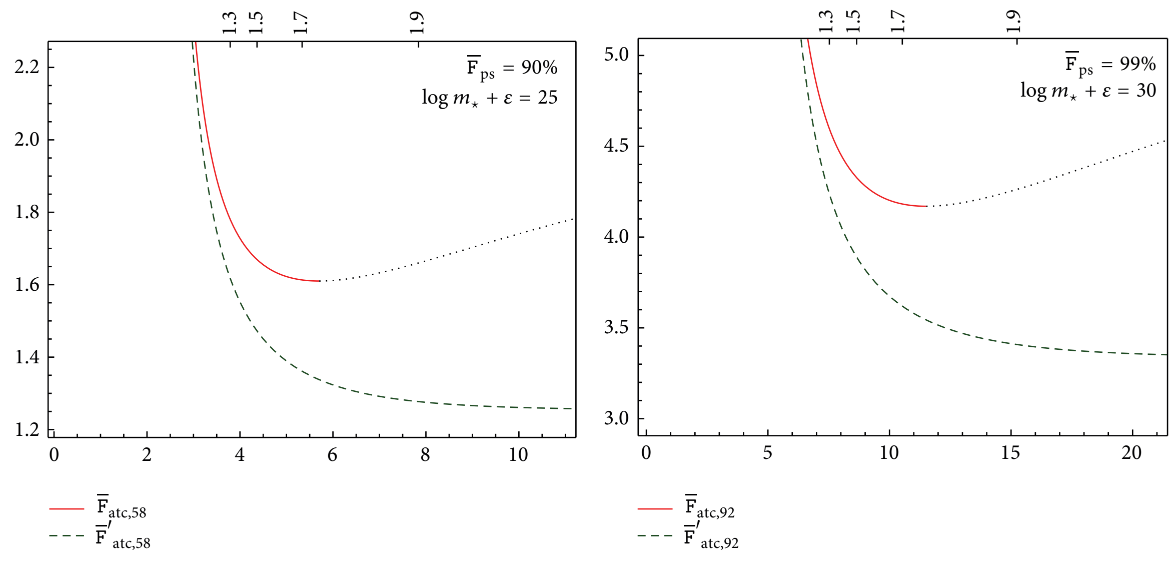
In every comparison box, the curve for the perfect fuzzy rainbow tradeoff appears much closer to the lower left corner than the points for the perfect rainbow tradeoff. The perfect fuzzy rainbow tradeoff can attain better online efficiency than the perfect rainbow tradeoff for the same online cost and attain equal online efficiency at lower precomputation cost. Furthermore, the perfect fuzzy rainbow tradeoff can attain online efficiency that is not possible with the perfect rainbow tradeoff.

Similar statements are true concerning the comparison between the perfect and nonperfect fuzzy rainbow tradeoffs, except that, at the lower end of the possible precomputation cost range, the two curves cross over each other, and the nonperfect fuzzy rainbow tradeoff can provide better online efficiency for the same precomputation effort. However, this is the region where the online efficiency is extremely bad, so that these parameters would be of limited practical interest.

In all, we can state that the perfect fuzzy rainbow tradeoff displays better performance than both the perfect rainbow tradeoff and the nonperfect fuzzy rainbow tradeoff, over a wide range of tradeoff algorithm application situations.

## 5. Conclusion

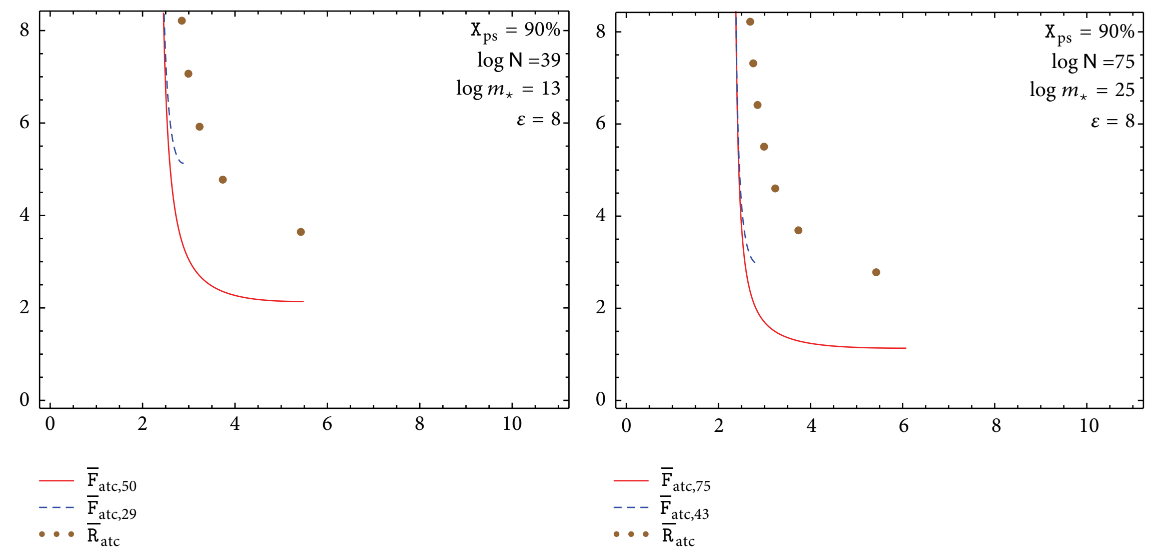
The execution behavior of the perfect table version of the fuzzy rainbow tradeoff algorithm was analyzed in this



(a)

(b)

Figure 2: The adjusted tradeoff coefficient $\overline{F}_{atc,s}$ and its simplified version $\overline{F}'_{atc,s}$ in relation to $\overline{F}_{pc,s}$ (bottom: $\overline{F}_{pc,s}$; left: $\overline{F}_{atc,s}$ and $\overline{F}'_{atc,s}$; top: $\overline{F}_{msc}$).



(a)

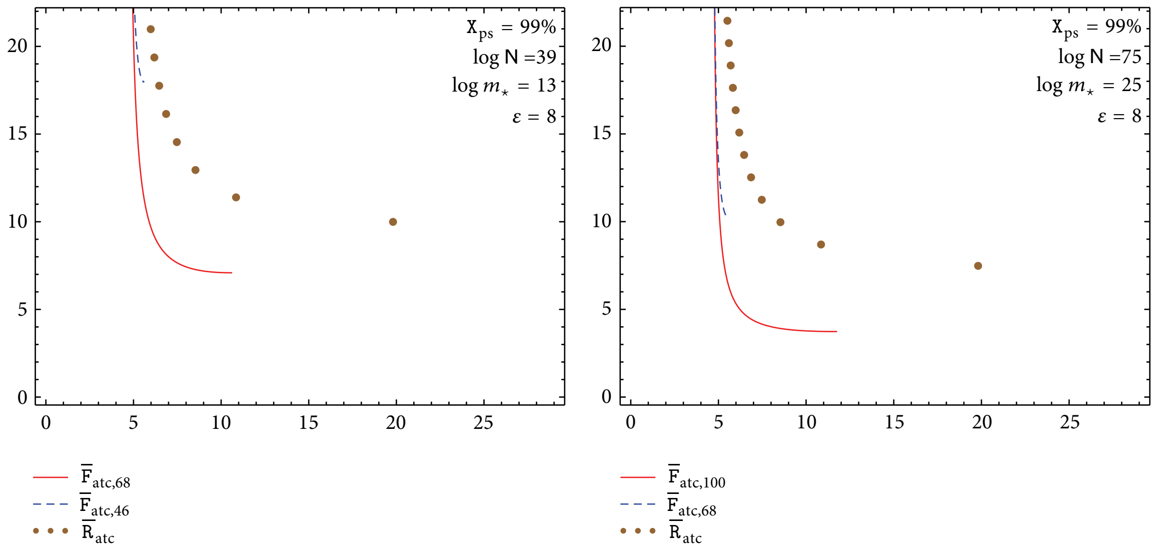
(b)

Figure 3: The adjusted tradeoff coefficients in relation to the precomputation coefficients for the perfect fuzzy rainbow, perfect rainbow, and nonperfect fuzzy rainbow tradeoffs at 90% success rate (bottom: $\mathtt{X}_{pc}$; left: $\mathtt{X}_{atc}$).

paper. The average case online computational complexity that fully accounts for the effects of false alarms was accurately obtained. The expected number of precomputation table entries and the number of bits required to record each table entry were also obtained accurately.

The results of our complexity analysis were used to compare the performance of the perfect fuzzy rainbow tradeoff with those of other tradeoff algorithms. The perfect rainbow tradeoff was recently argued by [13] to be advantageous over all other widely known tradeoff algorithms, and our recent work [9] had shown that the less widely known nonperfect fuzzy rainbow tradeoff outperforms the perfect rainbow tradeoff under certain circumstances. Hence, our comparison targets were set to the perfect rainbow tradeoff and the nonperfect fuzzy rainbow tradeoff.

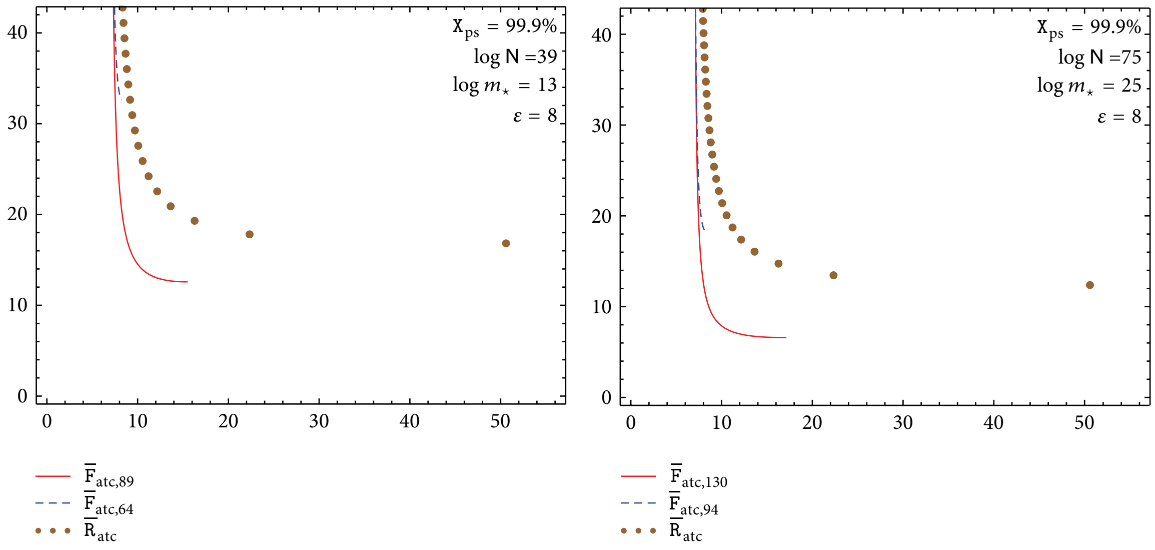
The comparison took the following aspects of the tradeoff algorithms into account: success rate of inversion, precomputation complexity, computational complexity of the online phase, and physical storage size of the precomputation table. The comparison called for a carefully designed rule that



(a)

(b)

FIGURE 4: The adjusted tradeoff coefficients in relation to the precomputation coefficients for the perfect fuzzy rainbow, perfect rainbow, and nonperfect fuzzy rainbow tradeoffs at 99% success rate (bottom: $\mathtt{X}_{\mathrm{pc}}$; left: $\mathtt{X}_{\mathrm{atc}}$).



(a)

(b)

FIGURE 5: The adjusted tradeoff coefficients in relation to the precomputation coefficients for the perfect fuzzy rainbow, perfect rainbow, and nonperfect fuzzy rainbow tradeoffs at 99.9% success rate (bottom: $\mathtt{X}_{\mathrm{pc}}$; left: $\mathtt{X}_{\mathrm{atc}}$).

correlated the number of bits per table entry to be used by each algorithm in a fair manner. The current work is the first to include the effects arising from the merge removal process of the perfect table creation into this rule.

We were able to conclude that the perfect fuzzy rainbow tradeoff is highly preferable to the perfect rainbow tradeoff. The perfect fuzzy rainbow tradeoff was also found to be superior to the nonperfect fuzzy rainbow tradeoff, except possibly at parameters that would have both of the algorithms performing very poorly during the online phase in exchange for a very small advantage in precomputation cost.

# Appendix

## Experimental Results

This section presents the results of four separate tests that support the theoretical findings of this paper. The first test shows the level of accuracy of the approximation claimed by Lemma 1. The subsequent two tests verify the correctness of two of our logical arguments that lie hidden within the proofs of technical lemmas. The final test verifies that our claim of online computational complexity is correct and serves as an overall checkup of our theory.

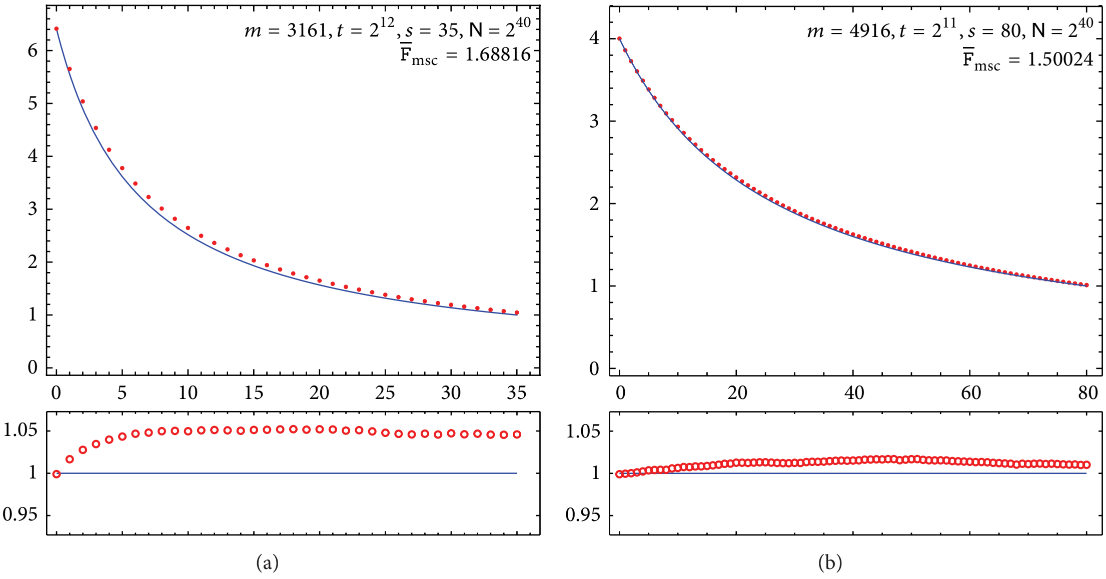


Figure 6: Number of $i$th color boundary points in a fuzzy rainbow matrix (*line*: theory; *dots*: test; *circles*: test/theory; $x$-axis: $i$; $y$-axis: $m_i/m$ and test/theory ratio).

The one-way function for the tests was taken to be the key to ciphertext mapping, under a randomly generated fixed plaintext, of AES-128. Independence between multiple tests was acquired through distinct randomly generated plaintexts. Truncations of 128-bit ciphertexts to binary strings of a certain fixed length and zero-padded extensions of these to 128-bit keys were used to bring the search space to a manageable size. 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. The reduction functions were set to constant XOR-ing operations.

Let us first check the accuracy of Lemma 1 through experiments. Recall that Lemma 1 is equivalent to (4) and that (4) was taken from our previous work [9], which dealt with the nonperfect fuzzy rainbow tradeoff. Note that the accuracy of (4) was already confirmed through tests in [18], even for small $s$ values. However, the parameter sets used there were such that the $\mathsf{F}_{\mathrm{msc}} = (m_0 t^2 s)/\mathsf{N}$ values were 0.92 and 1.37, and, according to (4), which is at least approximately correct, these correspond to $\overline{\mathsf{F}}_{\mathrm{msc}}$ values of 0.63 and 0.81. The results of Section 4.3 imply that our current interest should be with parameter sets belonging to the rough range $1.5 \le \overline{\mathsf{F}}_{\mathrm{msc}} \le 1.8$. Hence, we cannot rely on the previous test results to claim that the accuracy of Lemma 1 is sufficient for use with the perfect table case analysis.

One can infer from a careful review of how [9] obtained the approximate closed-form formula (4) from the iterative formula (3) that the inaccuracy of (4) is likely to increase as $s$ is made smaller and also as $\overline{\mathsf{F}}_{\mathrm{msc}}$ is made to approach 2. We experimented with multiple parameter sets for which the $(s, \overline{\mathsf{F}}_{\mathrm{msc}})$ pair was close to one of those appearing in Table 1, that is, those that correspond to optimal online efficiencies for some success rate requirement. We discovered that the inaccuracy was greater with optimal parameter sets for the lower success rates and that the level of (in)accuracy was rather stable among parameter sets for the same success rate. We could also confirm that the accuracy increased when $\overline{\mathsf{F}}_{\mathrm{msc}}$ was made smaller under a fixed $s$ value.

Two of the test results are given by Figure 6. After choosing parameters $m$, $t$, $s$, and $\mathsf{N}$, we computed $m_0$ through Lemma 1 and generated the fuzzy rainbow matrix from $m_0$ starting points. A total of ten fuzzy rainbow matrices were generated for each parameter set, and the number of $i$th color boundary points was recorded and averaged separately for each $i$. A small number of chains did not reach a DP within our chain length bound of $15t$, at various colors, and we discarded these without replacing them with newly generated chains. The lines of Figure 6 represent the theory given by Lemma 1 and the dots correspond to the experimental data. Each dot gives the count of the $i$th color boundary points, averaged over ten tests.

The test results for the worst parameter set we had experimented with are given in the two left-hand side boxes of Figure 6. This is the situation where our theory is least accurate, but even in this case, the bottom box shows that the largest inaccuracy of (4) is approximately 5%. The right-hand side boxes present test results for a parameter set whose $(s, \overline{\mathsf{F}}_{\mathrm{msc}})$ pair does not necessarily correspond to optimal online efficiency for any success rate. Since precomputation cost considerations will make parameter sets of suboptimal online efficiency more practical, the right-hand side boxes present the situation one would be experiencing in practice. The test results match our theory reasonably well, although not perfectly. We may conclude that Lemma 1 predicts the number of $i$th color boundary points sufficiently accurately for use in practice.

We wish to remark that our test results are in almost perfect agreement with what can be computed through the iterative formula (3), which (4) is supposed to approximate. Since $s$ values of interest are of manageable sizes, one could always revert to using the iterative formula (3) should there be the need for higher accuracy.

Let us now explain our second experiment. One argument that was crucial during our analysis was that the selection process of DP subchains $\overline{\mathtt{DM}}_i$ from among the DP subchains $\widetilde{\mathtt{DM}}_i$ is not correlated with the lengths of the DP subchains in $\widetilde{\mathtt{DM}}_i$. This claim is equivalent to the equation

$$\frac{\left|\overline{\mathtt{DM}}_i\right|}{m_s t} = \frac{\left|\widetilde{\mathtt{DM}}_i\right|}{m_i t} \tag{A.1}$$

that appeared during the proof of Lemma 3. This argument also allowed us to claim during the proof to Lemma 8 that $(\overline{\mathsf{F}}_{\mathrm{cr},1} + \cdots + \overline{\mathsf{F}}_{\mathrm{cr},i})t$ iterations of the one-way function are necessary to regenerate a precomputation chain up to the $i$th color. This is a new argument that had not been used in any of the previous works, and it would be reasonable to verify this claim experimentally.

Note that Lemma 3 allows us to expect these values to be of $1 - \Theta(1/4s)$ order, implying that there are more discarded chains within each DP submatrix for smaller $s$ values. Hence, in testing (A.1), to increase the chances of discovering possible errors, one would not only choose parameters for which $m_i/m_s$ is large but also choose to use small $s$ values. Since the formula of Lemma 3, given in terms of notation (9), is guaranteed to be accurate only for $s$ values of interest, which are not very small, it would be more appropriate to test our argument directly through (A.1), rather than through the formula of Lemma 3.

Test results for the two small $s = 8$ and $s = 15$ values are given in Table 2. The other parameters $m_0$ and $t$ were chosen so that the corresponding experimentally obtained $\overline{\mathsf{F}}_{\mathrm{msc}} = (m_s t^2 s)/\mathsf{N}$ values would be large, corresponding to a large $m_0/m_s$ ratio, while still falling within our range of interest. Tests for each of the two parameter sets were repeated ten times. All figures displayed by Table 2, other than the parameters, including the $m_s$ and $m_s t^2 s/\mathsf{N}$ values, are averages taken over the ten repetitions. In particular, the stated $(|\widetilde{\mathtt{DM}}_i|/m_i t)/(|\overline{\mathtt{DM}}_i|/m_s t)$ values are averages and not the simple ratios of the two averages appearing above these values. The experimental data strongly supports the correctness of (A.1).

Our third experiment aimed to test how accurate (35) was in presenting the cost of resolving a possible alarm associated with an online chain that is generated from the $i$th color. This was of particular interest because the formula depended on Lemma 9, which was stated as an approximation. The experimental verification of (35) would also increase our confidence in the extremely delicate random function arguments we gave during the proof of Lemma 8.

The results of our tests that were carried out with two separate sets of parameters are given in Table 3. Each set of data involves 50 precomputation tables created from the specified $m_0$-many starting points. For each precomputation table and each fixed starting color $i$, we generated sufficiently many online chains so as to observe approximately 2000 merges. Our test results are in good agreement with our theoretical predictions given by (35).

We have added another row of theoretical predictions in Table 3. Tracing back through the proofs of Lemmas 7, 8, and 9 and referring to Lemmas 1 and 3, one can see that (35) is a simplified expression for

$$\begin{aligned} t \Bigg\{ & \sum_{k=1}^{i} \frac{2}{(m_k t^2/\mathsf{N})} \ln\left(1 + \frac{m_k t^2/\mathsf{N}}{2}\right) \\ & \times \left(1 - \prod_{k=i}^{s} \frac{1}{1 + m_k t^2/\mathsf{N}}\right) \\ & + \left(1 - \frac{1}{1 + m_i t^2/\mathsf{N}}\right) \frac{m_s}{m_i} \Bigg\}. \end{aligned} \tag{A.2}$$

Since Lemma 1 is a closed-form formula that approximates the iterative formula (3), our theoretical predictions of the alarm cost can be made more accurate through (A.2) and (3), although the required calculations can be slightly uncomfortable due to the iterative nature of (3). Indeed, we can see in Table 3 that the theoretical predictions made through (A.2) and (3) are even closer to the test results than the predictions of the closed-form formula (35).

Finally, we present an experimental verification of Theorem 11, which is essentially equivalent to the claim that (38) gives the online computational complexity of the perfect fuzzy rainbow tradeoff. Since this test was meant to be an overall sanity check of our theory, we used realistic $s$ values.

For each choice of parameters $m$, $t$, $s$, and $\ell$, we computed $m_0$ through Lemma 1 and generated a full set of $\ell$ precomputation tables, each from $m_0$ starting points. After the completion of each precomputation phase, we generated $10^4$ random inversion targets and performed the online phase. Test results corresponding to three separate parameter sets are displayed in Table 4. During both the precomputation and the online phases, we discarded any chain that reached the length of $15t$ without a DP within any of its subchains. The computational effort associated with these discarded chains is included in the test online complexities.

Let us take a closer look at the first parameter set. Noting that $\log m = 21.0$, we truncated each ending point to the length of $\log|\mathrm{ep}| = 26$ bits, allowing for 5.0 extra bits of information. A small fraction of the ending points were made identical to each other, and we were careful to process all matching truncated ending points during the online phase. An 18-bit index is reasonable for the $\log m = 21.0$ situation, and its application to our truncated 26-bit ending points would allow each ending point to be stored in $\varepsilon = 3.0 + 5.0 = 26–18$ bits. However, we did not do so in favor of easier implementation and simply allocated 3 bytes and 4 bytes to the starting and ending points, respectively. Since $\log(m/s) = 15.2$, we are dealing with the $\log m_\star + \varepsilon \approx 23$ situation, and we can check through Table 1 that Test-1 used a parameter set for the 90% success rate that is close to optimal in view of the online phase.

Our Test-2 implementation recorded 24 bits of each ending point and this would correspond to $\log m_\star + \varepsilon = 13.1 + 24 - 16 \approx 21.1$, when a 16-bit index is used.

Table 2: Experimental confirmation that selection of $\overline{\mathtt{DM}}_i$ from $\widetilde{\mathtt{DM}}_i$ does not disturb average chain length.

$m_0 = 14000$, $t = 2^{13}$, $s = 8$, $\mathsf{N} = 2^{40}$; $m_s = 3604$, $\dfrac{m_s t^2 s}{\mathsf{N}} = 1.7598$

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $\dfrac{m_i}{m_s}$ | 2.9270 | 2.3391 | 1.9370 | 1.6338 | 1.4159 | 1.2461 | 1.1105 | 1.0000 |
| $\dfrac{\lvert\widetilde{\mathtt{DM}}_i\rvert}{m_i t}$ | 0.8649 | 0.8895 | 0.9090 | 0.9157 | 0.9293 | 0.9370 | 0.9446 | 0.9550 |
| $\dfrac{\lvert\overline{\mathtt{DM}}_i\rvert}{m_s t}$ | 0.8698 | 0.8923 | 0.9130 | 0.9144 | 0.9289 | 0.9369 | 0.9455 | 0.9550 |
| $\dfrac{\lvert\widetilde{\mathtt{DM}}_i\rvert}{m_i t} / \dfrac{\lvert\overline{\mathtt{DM}}_i\rvert}{m_s t}$ | 0.9947 | 0.9970 | 0.9956 | 1.0014 | 1.0005 | 1.0002 | 0.9991 | 1.0000 |

$m_0 = 40000$, $t = 2^{12}$, $s = 15$, $\mathsf{N} = 2^{40}$; $m_s = 7821$, $\dfrac{m_s t^2 s}{\mathsf{N}} = 1.7901$

| $i$ | 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 |
|---|---|---|---|---|---|---|---|---|
| $\dfrac{m_i}{m_s}$ | 4.1159 | 2.8944 | 2.2258 | 1.8053 | 1.5029 | 1.2921 | 1.1257 | 1.0000 |
| $\dfrac{\lvert\widetilde{\mathtt{DM}}_i\rvert}{m_i t}$ | 0.8942 | 0.9218 | 0.9376 | 0.9472 | 0.9567 | 0.9615 | 0.9634 | 0.9702 |
| $\dfrac{\lvert\overline{\mathtt{DM}}_i\rvert}{m_s t}$ | 0.8916 | 0.9223 | 0.9373 | 0.9445 | 0.9575 | 0.9604 | 0.9630 | 0.9702 |
| $\dfrac{\lvert\widetilde{\mathtt{DM}}_i\rvert}{m_i t} / \dfrac{\lvert\overline{\mathtt{DM}}_i\rvert}{m_s t}$ | 1.0029 | 0.9995 | 1.0004 | 1.0029 | 0.9992 | 1.0011 | 1.0004 | 1.0000 |

Table 3: Experimental verification of theoretically obtained expected cost of resolving a possible alarm associated with an online chain that starts from the $i$th color.

$m_0 = 32000$, $t = 2^{10}$, $s = 70$, $\mathsf{N} = 2^{38}$; $\text{Avg}(m_s) = 6179.5$, $\dfrac{\text{Avg}(m_s)t^2 s}{\mathsf{N}} = 1.6501$

| $i$ | 1 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|
| Test | 981.91 | 9144.6 | 16700. | 21868. | 23781. | 21544. | 14513. | 1653.9 |
| (35) | 975.69 | 9094.8 | 16575. | 21636. | 23470. | 21279. | 14263. | 1626.6 |
| (A.2) and (3) | 978.26 | 9136.9 | 16696. | 21850. | 23759. | 21588. | 14499. | 1656.6 |
| Test/(35) | 1.0064 | 1.0055 | 1.0076 | 1.0107 | 1.0132 | 1.0125 | 1.0176 | 1.0168 |
| Test/(A.2) and (3) | 1.0037 | 1.0008 | 1.0002 | 1.0008 | 1.0009 | 0.9980 | 1.0010 | 0.9984 |

$m_0 = 27000$, $t = 2^{11}$, $s = 100$, $\mathsf{N} = 2^{40}$; $\text{Avg}(m_s) = 4461.4$, $\dfrac{\text{Avg}(m_s)\, t^2 s}{\mathsf{N}} = 1.7019$

| $i$ | 1 | 10 | 20 | 30 | 50 | 70 | 90 | 100 |
|---|---|---|---|---|---|---|---|---|
| test | 1963.2 | 18961. | 36171. | 50616. | 67819. | 64260. | 32306. | 3462.6 |
| (35) | 1970.9 | 18878. | 35910. | 50239. | 67358. | 63441. | 31745. | 3377.4 |
| (A.2) and (3) | 1974.3 | 18932. | 36069. | 50539. | 67955. | 64167. | 32181. | 3427.2 |
| Test/(35) | 0.9961 | 1.0044 | 1.0073 | 1.0075 | 1.0069 | 1.0129 | 1.0177 | 1.0252 |
| Test/(A.2) and (3) | 0.9944 | 1.0016 | 1.0028 | 1.0015 | 0.9980 | 1.0014 | 1.0039 | 1.0103 |

Intending to reach a 95% success rate, we used $s = 56$, which is close to the optimal value for this situation. However, the $m$ and $t$ parameters were chosen so that the $\overline{\mathsf{F}}_{\text{msc}}$ value is smaller than what is optimal for the online phase. This parameter set should be more practical in view of lower precomputation cost.

Parameters for Test-3 were likewise chosen to be realistic rather than to be optimal for the online phase. One difference with the parameter set for Test-2 is that we truncated the ending points to a slightly longer length. We could verify that our prediction of the cost of resolving alarms was more accurate for this case than Test-1 and Test-2.

The theory and experimental figures for the three tests are in good agreement. In reality, as can be expected from Figure 6, the average $m_i$ values from the tests are slightly larger than our predictions and this brings about a success

Table 4: Experimental verification of the online time complexity as given by (38).

(a)

| $\mathsf{N} = 2^{40}$ | $m$ | $t$ | $s$ | $\ell$ | $\overline{\mathsf{F}}_{\text{msc}}$ | $m_0$ | $\log m_0$ | $\log m$ | $\log m_\star$ |
|---|---|---|---|---|---|---|---|---|---|
| Test-1 | 2078517 | $2^7$ | 56 | 173 | 1.73445 | 15654415 | 23.9 | 21.0 | 15.2 |
| Test-2 | 489178 | $2^8$ | 56 | 477 | 1.63281 | 2664424 | 21.3 | 18.9 | 13.1 |
| Test-3 | 1258291 | $2^7$ | 80 | 396 | 1.50000 | 5033162 | 22.3 | 20.3 | 13.9 |

(b)

| | Theory | | Test | | | | Reasonable $\varepsilon$ |
|---|---|---|---|---|---|---|---|
| | $\overline{\mathsf{F}}_{\text{ps}}$ | $T/t\ell s$ | $\log\lvert\text{sp}\rvert$ | $\log\lvert\text{ep}\rvert$ | $\overline{\mathsf{F}}_{\text{ps}}$ | $T/t\ell s$ | |
| Test-1 | 0.9002 | 12.3627 | 24 | 26 | 0.9048 | 12.6569 | 8 = 3.0 + 5.0 |
| Test-2 | 0.9501 | 9.0359 | 22 | 24 | 0.9571 | 9.1363 | 8 = 2.9 + 5.1 |
| Test-3 | 0.9900 | 6.8393 | 23 | 28 | 0.9907 | 6.8207 | 11 = 3.3 + 7.7 |

rate that is higher than expected. The higher success rate lowers the cost of generating online chains, while application of the ending point truncation technique raises the cost of resolving false alarms. The combination of the two opposite effects is what we are seeing in Table 4. We have verified through separate computations that replacing Lemma 1 with its iterative counterpart (3) produces even better predictions of at least the costs of generating the online chains. Hence, the small discrepancies between theory and test found in Table 4 are due to the accuracy limitations of Lemma 1 rather than to any oversights in our theoretical arguments.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIP) (NRF-2012R1A1B4003379).

## References

[1] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[2] D. E. Denning, *Cryptography and Data Security*, Addison-Wesley, Reading, Mass, USA, 1982.

[3] P. Oechslin, "Making a faster cryptanalytic time-memory trade-off," in *Advances in Cryptology—CRYPTO 2003*, vol. 2729 of *Lecture Notes in Computer Science*, pp. 617–630, Springer, Berlin, Germany, 2003.

[4] E. P. Barkan, *Cryptanalysis of Ciphers and Protocols [Ph.D. thesis]*, Technion—Israel Institute of Technology, 2006.

[5] E. Barkan, E. Biham, and A. Shamir, "Rigorous bounds on cryptanalytic time/memory tradeoffs," in *Advances in Cryptology—CRYPTO 2006*, vol. 4117 of *Lecture Notes in Computer Science*, pp. 1–21, Springer, Berlin, Germany, 2006.

[6] K. Nohl, "Attacking phone privacy," in *Proceedings of the Black Hat USA*, Las Vegas, Nev, USA, July 2010.

[7] K. Nohl and C. Paget, "GSM-SRSLY?" in *Proceedings of the 26th Chaos Communication Congress*, Berlin, Germany, December 2009.

[8] F. van den Broek and E. Poll, "A comparison of time-memory trade-off attacks on stream ciphers," in *Progress in Cryptology—AFRICACRYPT 2013*, vol. 7918 of *Lecture Notes in Computer Science*, pp. 406–423, Springer, Berlin, Germany, 2013.

[9] B. I. Kim and J. Hong, "Analysis of the non-perfect table fuzzy rainbow tradeoff," in *Information Security and Privacy*, vol. 7959 of *Lecture Notes in Computer Science*, pp. 347–362, Springer, New York, NY, USA, 2013.

[10] A. Biryukov, S. Mukhopadhyay, and P. Sarkar, "Improved time-memory trade-offs with multiple data," in *Selected Areas in Cryptography*, vol. 3897 of *Lecture Notes in Computer Science*, pp. 110–127, Springer, Berlin, Germany, 2006.

[11] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Advances in Cryptology—ASIACRYPT 2000*, vol. 1976 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, Berlin, Germany, 2000.

[12] J. Hong and S. Moon, "A comparison of cryptanalytic tradeoff algorithms," *Journal of Cryptology*, vol. 26, no. 4, pp. 559–637, 2013.

[13] G. W. Lee and J. Hong, "A comparison of perfect table cryptanalytic tradeoff algorithms," Cryptology ePrint Archive 2012/540, 2012.

[14] G. Avoine, P. Junod, and P. Oechslin, "Characterization and improvement of time-memory trade-off based on perfect tables," *ACM Transactions on Information and System Security*, vol. 11, no. 4, article 17, pp. 1–17, 2008, Preliminary version presented at INDOCRYPT 2005.

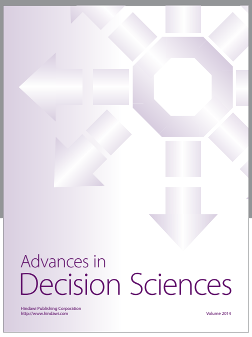
[15] A. Biryukov, A. Shamir, and D. Wagner, "Real time cryptanalysis of A5/1 on a PC," in *Fast Software Encryption*, vol. 1978 of *Lecture Notes in Computer Science*, pp. 1–18, Springer, Berlin, Germany, 2001.

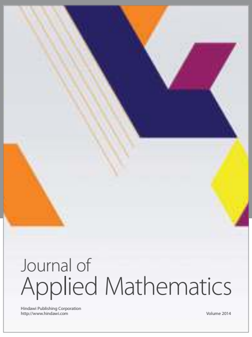
[16] J. Borst, *Block Ciphers: Design, Analysis, and Side-Channel Analysis [Ph.D. thesis]*, Katholieke Universiteit Leuven, 2001.

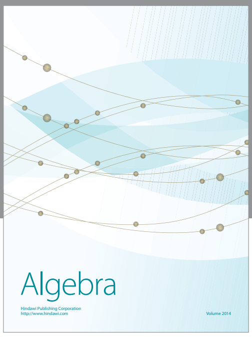
[17] F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, and J.-D. Legat, "A time-memory tradeoff using distinguished points: new analysis & FPGA results," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 593–609, Springer, Berlin, Germany, 2002.

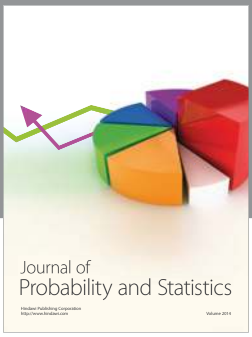
[18] B. I. Kim and J. Hong, "Analysis of the non-perfect table fuzzy rainbow tradeoff," Cryptology ePrint Archive 2012/612, 2012.

Advances in
Operations Research

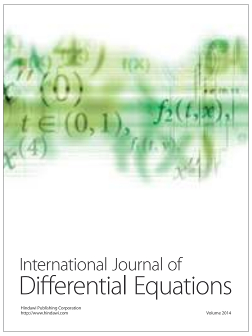

Advances in
Decision Sciences


Journal of
Applied Mathematics

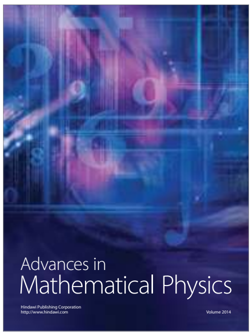

Algebra

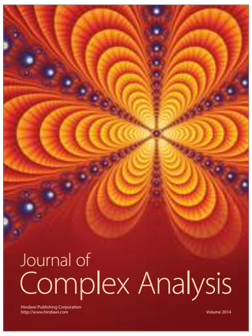

Journal of
Probability and Statistics

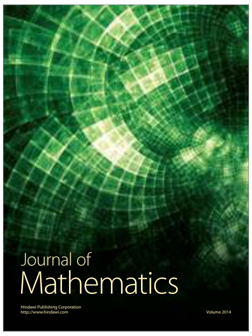
The Scientific
World Journal

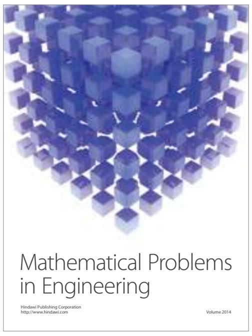

International Journal of
Differential Equations

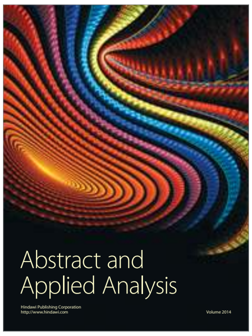
International Journal of
Combinatorics

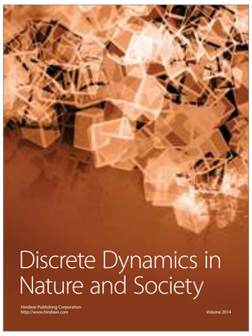

Advances in
Mathematical Physics

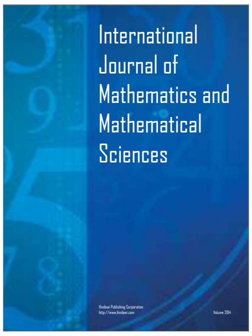

Journal of
Complex Analysis


Journal of
Mathematics


Mathematical Problems
in Engineering

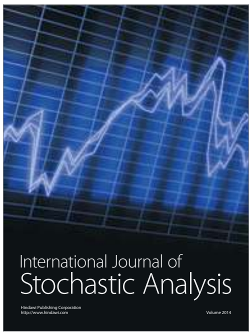

Abstract and
Applied Analysis


Discrete Dynamics in
Nature and Society


International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces


International Journal of
Stochastic Analysis

Journal of
Optimization