**Noname manuscript No.**
(will be inserted by the editor)

# A Comparison of Perfect Table Cryptanalytic Tradeoff Algorithms

**Ga Won Lee** · **Jin Hong**

February 6, 2013

**Abstract** The performances of three major time memory tradeoff algorithms were compared by a recent paper. The algorithms considered there were the classical Hellman tradeoff and the non-perfect table versions of the distinguished point method and the rainbow table method. This paper adds the perfect table versions of the distinguished point method and the rainbow table method to the list, so that all the major tradeoff algorithms may now be compared against each other.

Even though there are existing claims as to superiority of one tradeoff algorithm over another algorithm, the algorithm performance comparisons provided by the current and preceding papers are of more practical value. Comparisons that take both the cost of pre-computation and the efficiency of the online phase into account, at parameters that achieve a common success rate, can now be carried out with ease. Comparisons can be based on the expected execution complexities rather than the worst case complexities, and details such as the effects of false alarms and various storage optimization techniques need no longer be ignored.

A large portion of this paper is allocated to accurately analyzing the execution behavior of the perfect table distinguished point method. In particular, we obtain a closed-form formula for the average length of chains associated with a perfect distinguished point table.

**Keywords** time memory tradeoff · distinguished point · rainbow table · perfect table · algorithm complexity

## 1 Introduction

A cryptanalytic time memory tradeoff algorithm is a method for inverting one-way functions with the help of pre-computed data. It is widely used today by hackers and also during criminal investigations to recover passwords from the knowledge of the password hash. In

J. Hong
Department of Mathematical Sciences and ISaC, Seoul National University, Seoul 151-747, Korea
E-mail: jinhong@snu.ac.kr

G. W. Lee
Department of Mathematical Sciences and ISaC, Seoul National University, Seoul 151-747, Korea
E-mail: gwlee87@snu.ac.kr

the pre-computation phase, massive computation specific to the one-way function of interest is performed and a compact digest of the obtained information is stored as tables. When the target image for inversion is given, computation that utilizes the pre-computed tables is performed to recover the pre-image with some probability, and this part is referred to as the online phase.

The execution behavior of any tradeoff algorithm can be manipulated through its many parameters. Existing analyses show that most tradeoff algorithms satisfy the tradeoff curve

$$TM^2 = c\,\mathsf{N}^2, \tag{1}$$

for some small constant $c$, where $T$ is the online execution time, $M$ is the size of the memory space required to store the pre-computation tables, and $\mathsf{N}$ is the size of the space the one-way function is acting on. This means that if a tradeoff algorithm executes in time $T$ using tables of combined size $M$ under some set of its parameters, then given any other $T'$ and $M'$ such that $TM^2 = T'M'^2$, there exists another set of parameters under which the algorithm will execute in time $T'$ using storage $M'$. Thus, each algorithm allows tradeoffs to be made between the online execution time and the storage requirement.

There are many time memory tradeoff algorithms available today, with most of them having roots in the classical algorithm by Hellman [7]. The most widely known algorithms are the distinguished point variant of the Hellman's original algorithm [4, 5] and the rainbow table method [12], which we shall refer to in this paper as the DP tradeoff and the rainbow tradeoff, respectively. Both of these algorithms have two subversions that work with the non-perfect tables and the perfect tables.

Comparison of tradeoff algorithm performances has been a controversial subject, with every newly announced algorithm claiming superiority over existing algorithms. The difficulty in accurately analyzing the execution behavior of these algorithms is clearly one reason for this confusion, but another source has been the absence of an acceptable method for numerically presenting the performances of tradeoff algorithms in a manner that closely reflects our intuition concerning their relative usefulness or practicality.

Let us take a moment to explain a reasonable method of tradeoff algorithm comparison that was recently suggested by [11]. Notice that the tradeoff curve (1) corresponding to any specific tradeoff algorithm presents the complete list of $(T,M)$-pair options that are made available by the algorithm. Thus the tradeoff curve expresses the required online resources or the online execution behavior of an algorithm completely, and one may accept the *tradeoff coefficient* $c = \frac{TM^2}{\mathsf{N}^2}$ as a good measure of how efficiency an algorithm is, with a smaller coefficient indicating a more efficient algorithm. Indeed, many previous claims as to the superiority of one algorithm over another have focused on this value.

However, the tradeoff coefficient alone cannot fully capture our intuition of how good an algorithm is. Since it is obvious that the online time can always be reduced if one is willing to accept a lower success rate of inversion, the tradeoff coefficient must change with the requirement on the inversion success rate. Furthermore, even when one is aiming for a fixed success rate, it is only reasonable to anticipate a more efficient online phase, or, equivalently, a smaller tradeoff coefficient, after a larger investment in the pre-computation phase.

In short, the online efficiency of each algorithm can be expressed succinctly through the tradeoff coefficient, but each algorithm allows further tradeoffs to be performed between the online efficiency, pre-computation effort, and success rate, while what we intuitively feel as the practicality or usefulness of an algorithm is directly connected to the overall behavior of the algorithm concerning these upper level tradeoffs. The difficulty of algorithm comparison lies in that, unlike the tradeoffs between time and storage that may commonly be expressed

in the form (1) for most tradeoff algorithms, the equations that express the tradeoffs between the three aforementioned factors are very different among the major tradeoff algorithms. Hence, no single numeric value that can be computed for all algorithms in a common manner is likely to capture the performances of the algorithms concerning the upper level tradeoffs.

The solution suggested by [11] is to let the algorithm implementers make the final judgement and choice based on their requirements, available pre-computation and online resources, and personal taste, and to only present the information necessary for this decision in a coherent manner. Parameters for different algorithms are first restricted to those that achieve the same success rate. Then the tradeoffs between pre-computation cost and tradeoff coefficient are presented as curves for each algorithm. Each curve represents the complete list of options provided by one algorithm as to what degree of online efficiency can be obtained after a certain amount of pre-computation investment, at the specified success rate. Implementers that place different relative values on the online efficiency and the pre-computation cost will choose to use different algorithms. In fact, comparisons of the algorithms themselves are no longer meaningful, and each implementer will choose an algorithm together with the online efficiency and pre-computation cost pair made available by that algorithm, based on his or her favored balance between the two factors, from among all the options made available by all the algorithms.

The work [11] first computes the success rates, pre-computation costs, and accurate tradeoff coefficients for the classical Hellman, non-perfect DP, and non-perfect rainbow tradeoffs. These complexities and properties are presented as functions of the algorithm parameters. Then, after fixing a small number of specific success rates of interest, parameters are restricted to those achieving these success rates, and the upper level tradeoffs between the pre-computation cost and tradeoff coefficient are presented as curves, separately for each algorithm. After carefully adjusting the units expressing the tradeoff coefficients for the three algorithms into one directly comparable unit, the three curves were superimposed into one graph. This comprehensive and coherent display of information allows for someone considering the use of the tradeoff algorithms to decide on the most desirable balance between pre-computation investment and online efficiency from among the numerous options made available by the three tradeoff algorithms, at any required success rate.

The current paper completes the task started in [11] by dealing with the two remaining major tradeoff algorithms, namely, the perfect DP and perfect rainbow tradeoffs. We compute the tradeoff coefficients for these two algorithms and present the upper level tradeoffs between pre-computation cost and online efficiency as graphs at a small number of fixed success rates. Similar graphs for any other success rate may easily be obtained from our formulas. An overly simplified conclusion that may be drawn from the graphs is that, under typical situations, the perfect rainbow tradeoff is likely to be preferred over the other four algorithms that have been mentioned. However, as we have discussed, the final judgement is not ours to make, and may be different under each specific situation.

Since this work is a direct extension of the work [11], we shall not repeat the contents of [11] that advocate the subject of our study. The readers are strongly urged to have at least a rough understanding of [11] before reading the current paper. In fact, it should be possible for the impatient reader with a full understanding of [11] to jump straight to Figure 2 and understand the core findings of this paper.

The rest of this paper is organized as follows. In the next section, we fix the terminology, clarify the exact versions of the algorithms we are analyzing, and review existing analyses of the perfect DP and perfect rainbow tradeoffs. Section 3 is devoted to fully analyzing the execution behavior of the perfect DP tradeoff, and is the most technical part of this paper. Analysis of the expected online time complexity that does not ignore false alarms

is given and the tradeoff coefficient is computed. The issue of storage optimization is also discussed and tests that give strength to the correctness of our theoretic developments are presented. The perfect rainbow tradeoff is treated in Section 4. There are previous analyses we can utilize and obtaining the tradeoff coefficient for the perfect rainbow tradeoff is much easier than with the perfect DP tradeoff. The information we have prepared concerning the perfect DP and perfect rainbow tradeoffs is presented in Section 5 as graphs that allow direct comparisons between different algorithms and also between different parameter sets for the same algorithm. Finally, the paper is summarized in Section 6. The appendices contain further discussions that could be of interest. In particular, we explain in Appendix C that the existing analyses of the perfect DP tradeoff were not accurate enough for the purpose of algorithm comparisons.

## 2 Preliminaries

In this section, after setting the grounds of our discussion, we review some of the existing related works. Only the theoretic developments concerning the accurate time and storage analyses of the perfect DP and the perfect rainbow tradeoffs are explained. Some of the contents we do not present would introduce other tradeoff algorithms or consider implementation issues. There are also theories concerning asymptotic complexity bounds [2] on a general class of tradeoff algorithms and analyses of the full costs [17] of many cryptographic attack algorithms. We acknowledge that even the papers we introduce contain much more content than what is explained here.

The reader is assumed to be familiar with the basics of the tradeoff technique. In particular, the explicit tradeoff algorithms will not be explained. If the reader wishes for a quick overview of the tradeoff techniques that includes brief descriptions of the classical Hellman, distinguished point, and rainbow tradeoff algorithms, it will be convenient to refer to [11], since the notation used here is compatible with that of [11]. The paper also clarifies many obscure technical details[1] that are not discussed elsewhere in the related literature, and which should be of interest to the mathematically oriented cryptographers.

### 2.1 Terminology, Notation, and Algorithm Clarification

Throughout this paper, the function $F : \mathscr{N} \to \mathscr{N}$ will always act on a set $\mathscr{N}$ of size $\mathsf{N}$ and the $k$-times iterated composition $F \circ \cdots \circ F$ of function $F$ is written as $F^k$. In practical applications, the function $F$ is the specific one-way function to be inverted, but it is treated as a random function during any theoretic analysis.

To reduce confusion, in this work, the word *efficiency* is always associated with an algorithm's competitiveness in the use of the online resources, whereas the ability to balance the online efficiency, the pre-computation cost, and sometimes also the success rate, against each other, is referred to with the word *performance*.

The approximation $(1-\frac{1}{b})^a \approx e^{-\frac{a}{b}}$, which is valid when $a = O(b)$, is used frequently throughout this paper without any explanation. A more precise statement of this approximation may be found in [11, Appendix A]. Infinite sums are also frequently approximated by

[1] Let us mention just one example. The objective of any tradeoff algorithm discussed in this work will be to recover the randomly chosen exact input that was used to create the given inversion target, rather than to recover any pre-image corresponding to the inversion target. This detail, which the previous sentence has *not* explained in full, is important when attempting an analysis of the accuracy aimed for in this work. However, even this most basic definition of a successful inversion is seldom made clear in the tradeoff literature.

appropriate definite integrals throughout this paper. Both kinds of approximations will be very accurate whenever we use them, as long as a reasonable set of parameters is used with the tradeoff algorithm, and will be written as equalities rather than as approximations.

Many parameters need to be fixed before any tradeoff algorithm can be put to use. Some of these are the chain length $t$, the number of rows or chains $m$ for each pre-computation table, and the number of tables $\ell$. When working with the DP tradeoff, we assume a distinguishing property which is satisfied by a random point of the search space $\mathscr{N}$ with probability $\frac{1}{t}$, so that the expected length of a random DP chain is $t$. When dealing with perfect tables, $m$ will denote the number of distinct ending points or the number of rows after removal of merging chains, rather than the number of all chains that were initially generated while preparing the pre-computation table.

In the DP tradeoff case, the parameters are usually chosen so that $mt^2 \approx \mathsf{N}$ and $\ell \approx t$. In the rainbow tradeoff case, it is more usual to have $mt \approx \mathsf{N}$ and a small number of tables $\ell$. We use notation $\bar{\mathtt{D}}_{\mathrm{msc}} = \frac{mt^2}{\mathsf{N}}$ for the perfect DP tradeoff and $\bar{\mathtt{R}}_{\mathrm{msc}} = \frac{mt}{\mathsf{N}}$ for the perfect rainbow tradeoff and refer to these values, which are assumed to be neither large nor very close to zero, as the matrix stopping constants. The corresponding value is written as $\mathtt{D}_{\mathrm{msc}}$ for the non-perfect DP tradeoff.

We distinguish between a pre-computation *table*, which consists of starting point and ending point pairs, and a pre-computation *matrix*, which is the collection of all chains associated with a pre-computation table.

The coverage rate $\bar{\mathtt{D}}_{\mathrm{cr}}$ of a perfect DP matrix is defined to be the expected number $|\bar{\mathtt{DM}}|$ of distinct nodes in a perfect DP matrix, divided by $mt$. More precisely, only the points that are used as inputs to the one-way function are counted, so that the ending point DPs are excluded in the count $|\bar{\mathtt{DM}}|$ and $\bar{\mathtt{D}}_{\mathrm{cr}}\frac{mt}{\mathsf{N}}$ is the success probability associated with a single perfect DP table. The coverage rate $\bar{\mathtt{R}}_{\mathrm{cr}} = \frac{|\bar{\mathtt{RM}}|}{mt}$ of a perfect rainbow matrix is defined in exactly the same way.

Since there are many variations to the DP tradeoff, let us clarify the exact version of the DP tradeoff that will be analyzed in this work.

- The perfect table case is treated. If chain merges are discovered while computing a DP matrix, all chains except for the longest one among any set of merging chains are discarded [4, 5].
- Any implementation of the DP tradeoff will introduce an upper bound $\hat{t}$ on the length of pre-computation and online chains to deal with chains falling into loops [4, 5]. A lower bound $\check{t}$ can also be used [13, 16] to discard short pre-computation chains that contribute little to the search space coverage. In this work, no lower bound and a sufficiently large upper bound on chain lengths are assumed. This simplifies our theoretic developments by ensuring that the possibility of an online chain not meeting the chain length bound conditions will be negligible, and also by allowing us to ignore the effects of discarding long or short pre-computation chains. A brief justification as to why treating just this case is sufficient is given in Appendix A.
- The work [4, 5] suggests that the chain lengths of each pre-computation chain and the maximum pre-computation chain length for each table be recorded in the DP table. However, the recording of individual chain lengths has a negative effect on the physical amount of required storage, and we can argue heuristically that the positive effect of the maximal chain length information is very limited. Neither suggestion is followed in this work.
- Sequential starting points, rather than random ones, are used [1, 3, 4]. Then, if $m_0$ chains were generated per table before removal of merges, each starting point can be recorded

in $\log m_0$ bits, which should be much smaller than the $\log \mathsf{N}$ bits required to record a random point.
- Knowledge of the distinguishing property makes certain parts of the ending point redundant. These parts are not recorded in the pre-computation table to save $\log t$ bits of storage per ending point [3].
- The ending points are truncated to a certain length before being written to storage [2, 3]. Since some ending point information is lost, this will increase the frequency of false alarms. However, the side effects of truncation can be maintained at a manageable level by controlling the degree of truncation. Details are discussed later in this work.
- The index file technique [3] is used in recording the pre-computation tables. This allows reduction of nearly $\log m$ further bits of storage per truncated ending point without any loss of ending point information.
- The online chain record [8, 15] technique is used. While generating an online chain, one keeps track of not just the current foremost point of the chain, but records all the generated intermediate points. When resolving an alarm, one compares the current end of the regenerated pre-computation chain against the complete online chain, rather than just the inversion target point, so that one may stop the pre-computation chain regeneration at the exact position of chain merge, rather than at the common terminal DP.
- The work [8, 15] suggests that all the pre-computation tables be processed in parallel, rather than sequentially, during the online phase. For the case of non-perfect DP tradeoff, this idea was shown to have a small positive effect [10]. However, the parallel version of the perfect DP tradeoff will not be analyzed in this work. Treatment of parallelization is outside the scope of this work, but we expect our work to become an important stepping stone for anyone interested in analyzing the parallel version. Some comments on this issue are given in Appendix A.

There are also possible variations to the rainbow tradeoff, and the version treated in this work is clarified below. All techniques that we mention below are analogs of techniques we have already described for the DP tradeoff.

- The perfect table case is treated. Only one chain among any set of merging chains is retained [12]. All chains are of identical length and the method of choosing which chain to retain is irrelevant to our analysis.
- Sequential starting points, rather than random ones, are used to reduce the storage requirements of the starting points.
- The ending points are truncated to an appropriate length, to be discussed later, before being written to storage.
- The index file technique is used to reduce $\log m$ further bits of storage per ending point.
- The small number of multiple rainbow tables are processed in parallel [12] during the online phase. This is not necessarily what is usually meant by the parallelization of an algorithms, in which case even the processing of each table would be shared by multiple processors. If the online phase must run on a single processor, the multiple tables can be processed in a round-robin fashion to simulate the parallel table processing.

Applications of the perfect table technique to the DP and rainbow tradeoffs are expected to increase both the online efficiency and the pre-computation cost. Hence, it is not clear if the benefit of using perfect tables outweighs its drawback. Providing information that can be used to settle this question is one of the objectives of this paper. Truncation of ending points must also be used carefully, since the storage reduction is associated with an increase in online time. However, all other techniques we are employing are only advantageous, when used appropriately in typical environments.

### 2.2 Existing Analyses of the Perfect DP Tradeoff

The book [6, p.100] gives credit to Rivest for first suggesting to apply the DP technique to the classical Hellman tradeoff, but no corresponding formal article was published. The first analysis of the DP tradeoff that attempts to take the non-uniform chain lengths of the DP matrix into account was given by [4, 5]. There, credit is given to the unpublished work [13] for also having studied the DP tradeoff independently.

Many interesting variables were introduced by [4, 5] while analyzing the perfect DP tradeoff. The first of these is the expected number of chains $\alpha$ after removal of merging chains. The average of DP chain lengths $\beta_0$ and $\bar{\beta}$, before and after removal of merging chains, respectively, were also introduced. ([4, 5] writes $\bar{\beta}$ as $\beta$.) Note that the variable $\alpha$ is equal to the parameter $m$ used in this paper, but the work [4, 5] treated the number of pre-computation chains to be computed before collision removal as a given preset parameter. The success probability and online time estimates for the perfect DP tradeoff were given as equations involving $\alpha$ and $\bar{\beta}$. They also stated certain relations satisfied by $\alpha$, $\beta_0$, $\bar{\beta}$, and some other variables. However, they were unable to derive formulas for computing $\alpha$ and $\bar{\beta}$ from the externally provided parameters. Furthermore, as was pointed out by [16], some of their arguments treated the merges of pre-computation chains inadequately and were problematic.

The subsequent work [16] gave a more advanced analysis of the perfect DP tradeoff. They started by computing $\beta_0$ for the case when the chain length bounds $\check{t}$ and $\hat{t}$ are both enforced. ([16] writes $\beta_0$ as $\beta$.) Then the number of distinct nodes expected in a perfect DP matrix was expressed using the variable $\beta_0$. Because $\check{t}$ and $\hat{t}$ were taken into consideration while computing the node count, the number of DP chains of any specified length range appearing in a perfect DP matrix could be extracted from the node counts by focusing on sub-matrices of the total DP matrix. The obtained information on the chain length distribution was then used in an ad hoc manner to compute $\bar{\beta}$. ([16] writes $\bar{\beta}$ as $\beta_{mod}$.) Finally, the distinct ending point count $\alpha$ was easily expressed as a function of the perfect matrix node count and $\bar{\beta}$.

Note that $\alpha$ and the node count for a perfect DP matrix are directly connected to the storage complexity and the success rate of the tradeoff algorithm, respectively. The paper also provides a simple argument concerning the pre-computation cost and an upper bound on the time complexity of the online phase.

The analysis of the perfect DP tradeoff given by [16] may seem rather complete, except that the effects of false alarms were disregarded during the time complexity analysis. Since we are also claiming to have done the same analysis, a comparison of results is given in Appendix C. Our observation is that the results of [16] are only valid as first approximations, and that these are too rough for the purpose of this paper.

The later work [1] also discusses the perfect DP tradeoff, but they only consider the special case when the DP matrix consists of the maximum number of non-merging DP chains that may be collected for a specified DP probability. However, during their analysis, they oddly assume that the starting points for these chains are DPs. In any case, their result concerning the success probability requires knowledge of the average chain length associated with the maximal perfect DP matrix, but they were unable to provide this value except through experiments. Furthermore, since increasing the number of non-merging rows reduces the average chain length and possibly even the search space coverage, it is unclear if maximal perfect DP tables can be associated with being optimal in any sense.

The final work we mention is [14]. There, a lot of effort was invested in deriving a formula for $\beta_0$, but their end result is almost identical to what may be found in [16]. ([14]

writes $\beta_0$ as $\beta$.) The formulas of [14] and [16] for $\beta_0$ will exhibit noticeable difference only when $\hat{t}$ is close to $\mathsf{N}$, which is unrealistically large. After reobtaining $\beta_0$, they derive a formula for $\alpha$ that depends on $\beta_0$, but the argument is very terse and their logic is not clear. Finally, the two variables $\alpha$ and $\beta_0$ are combined to give the success probability of the tradeoff algorithm, but they seem to be confusing the concepts of $\beta_0$ and $\bar{\beta}$ at this point.

### 2.3 Existing Analyses of the Perfect Rainbow Tradeoff

The introduction of the rainbow tradeoff [12] was accompanied with a rudimentary analysis, which included the worst case online time complexity. The worst case refers to when the online phase algorithm processes all the pre-computation tables without returning the correct answer. However, the effects of false alarms were not accounted for in this worst case complexity claim. They compared the worst case complexity against the similarly rough worst case complexity of the DP tradeoff and claimed that the rainbow tradeoff was more efficient by a factor of two. This was then combined with heuristic arguments, mainly concerning false alarms, for a claim in much higher advantage. Most of their arguments referred to the non-perfect rainbow tradeoff and the perfect table version made an appearance only at the end of the paper, but the complexity analyses provided were rough enough to be applicable to both versions.

A more serious analysis of the perfect rainbow tradeoff appeared in [1]. It treated the expected online time complexity, rather than the worst case complexity, and took the effects of false alarms into account. Their stated complexity results hold true only in the case of maximal perfect tables, but a large part of the proofs for these results are valid for any perfect rainbow table. Once the proofs are understood, it is rather straightforward to write out the corresponding statements for the general perfect rainbow tables. In fact, it does not seem possible to obtain later parts of their work that compare tradeoff algorithms, without knowledge concerning these non-maximal cases, but the details were not provided in the paper.

The expected online time complexity of the perfect rainbow tradeoff that does not ignore false alarms was also given by [9]. There the complexity results for the general perfect rainbow tables were stated as closed-form formulas. These are easier to use and manipulate than the formulas of [1], which were given as certain double summations that further involved iterative computations if the general perfect rainbow tables were to be considered. However, the results of [1] and [9] should agree accurately when numerically evaluated on any specific set of reasonable parameters.[2] Our theoretic developments concerning the perfect rainbow tradeoff will rely heavily on these results.

Concerning the success rate of the rainbow tradeoff, note that this is trivial to write down for the perfect table version [12]. A formula for the success rate of even the non-perfect rainbow tradeoff already appeared in [12]. However, iterative computations were required to evaluate the formula on any specific parameter set. A simple closed-form formula that can replace this iterative part, for the special case of $\mathsf{N}$ starting points, was presented in [1], while studying the success rate of the maximal perfect rainbow tradeoff. The closed-form formula was slightly modified in [9] to work for any non-perfect rainbow table and was used to study the online complexities of the rainbow tradeoff. The success rate of the non-perfect rainbow tradeoff is not used directly in this work, but plays a crucial role in studying

[2] The analyses of [1] and [9] extend further to the application of checkpoints [1] on the perfect rainbow tradeoff, where the two do not seem to be in agreement.

the behavior of false alarms in the perfect rainbow tradeoff, and the current work relies on previous results [1, 9] that have worked out these details.

Let us mention one more issue that is not necessarily specific to the perfect rainbow tradeoff, but is closely related to this work. The work [2] claimed that each entry in the pre-computation table for the DP tradeoff can be represented by half the number of bits required for the rainbow tradeoff, but their explanation was rather brief. They followed this claim with a short argument stating that, if the effects of false alarms were to be ignored, one must conclude that the DP tradeoff is twice as efficient as the rainbow tradeoff. An attempt to refute this was made by [1], which maintained that the claim of [2] concerning the required storage bits per table entry was incorrect. With neither [2] nor [1] providing any detail, the work [11] clarified that, in the case of non-perfect tradeoffs, the storage requirement comparison of [2] was correct, but that the rainbow tradeoff may still be seen as being advantageous over the DP tradeoff in typical environments. However, the case of the perfect tradeoffs was left untreated.

## 3 Perfect DP Tradeoff

In this section, we deal with the perfect DP tradeoff that uses a sufficiently large upper bound and no lower bound on the chain length. Before starting our main analysis, let us check how often one can expect to see over-length chains when using a sufficiently large chain length bound.

The probability for a chain generated from a given starting point to fall into an infinite loop without reaching a DP within its first $\hat{t}$ iterations is

$$\begin{aligned}\frac{1}{\mathsf{N}}+\Big(1-\frac{1}{t}-\frac{1}{\mathsf{N}}\Big)\frac{2}{\mathsf{N}}+\Big(1-\frac{1}{t}-\frac{1}{\mathsf{N}}\Big)\Big(1-\frac{1}{t}-\frac{2}{\mathsf{N}}\Big)\frac{3}{\mathsf{N}}+\cdots\cdots\\ \cdots\cdots+\Big(1-\frac{1}{t}-\frac{1}{\mathsf{N}}\Big)\Big(1-\frac{1}{t}-\frac{2}{\mathsf{N}}\Big)\cdots\Big(1-\frac{1}{t}-\frac{\hat{t}-2}{\mathsf{N}}\Big)\frac{\hat{t}-1}{\mathsf{N}},\end{aligned} \tag{2}$$

and the probability for a chain not to reach a DP within its first $\hat{t}$ iterations, without falling into a loop, is

$$\Big(1-\frac{1}{t}-\frac{1}{\mathsf{N}}\Big)\Big(1-\frac{1}{t}-\frac{2}{\mathsf{N}}\Big)\cdots\Big(1-\frac{1}{t}-\frac{\hat{t}-2}{\mathsf{N}}\Big)\Big(1-\frac{1}{t}-\frac{\hat{t}-1}{\mathsf{N}}\Big). \tag{3}$$

The sum of these two terms is the probability for a chain not to reach a DP within its first $\hat{t}$ iterations, and may be approximated by

$$\frac{1}{\mathsf{N}}+\Big(1-\frac{1}{t}\Big)\frac{2}{\mathsf{N}}+\cdots+\Big(1-\frac{1}{t}\Big)^{\hat{t}-2}\frac{\hat{t}-1}{\mathsf{N}}+\Big(1-\frac{1}{t}\Big)^{\hat{t}-1}, \tag{4}$$

under the condition $t\hat{t}\ll\mathsf{N}$. If we further assume that $\frac{\hat{t}}{t}$ is not *too* large, we may approximate the above once more with

$$\frac{t^2}{\mathsf{N}}\int_0^{\frac{\hat{t}}{t}}e^{-u}u\,du+e^{-\frac{\hat{t}}{t}}=\frac{t^2}{\mathsf{N}}\Big\{1-\Big(1+\frac{\hat{t}}{t}\Big)e^{-\frac{\hat{t}}{t}}\Big\}+e^{-\frac{\hat{t}}{t}}. \tag{5}$$

This probability approaches $\frac{t^2}{\mathsf{N}}=O\big(\frac{1}{m}\big)$ very quickly, as $\frac{\hat{t}}{t}$ is increased. For example, even at the moderately large chain length bound of $\hat{t}=15t$, the probability for a randomly generated chain to be discarded due to its length is $\frac{t^2}{\mathsf{N}}0.999995+\frac{3.05902}{10^7}$, which is small enough for our purposes. However, because of the first term, which corresponds to looping chains, the number of long chains expected during the generation of a full DP matrix cannot be made arbitrarily close to zero by increasing the chain length bound.

### 3.1 Online Efficiency

This is the most complicated part of this paper. We will derive formulas describing the success probability, pre-computation cost, and tradeoff coefficient of the perfect DP tradeoff. The discussion will require previous results concerning the non-perfect DP tradeoff.

Let us visualize a non-perfect DP matrix as having been aligned at the ending points and use $\overleftarrow{m}_k$ to denote the number of distinct points expected in its column that is $k$ iterations away from the ending points. In particular, $\overleftarrow{m}_0$ denotes the number of non-merging rows in the DP matrix.

**Lemma 1** *We have $\overleftarrow{m}_0 \approx \overleftarrow{m}_1$. In other words, in a non-perfect DP matrix, the number of distinct ending points may be approximated by the number of distinct points that are a single iteration away from these ending point DPs.*

*Proof* Given a set of $\overleftarrow{m}_1$ points, which are known to be a single $F$-iteration away from the DPs, the size of its $F$-image is expected to be

$$\begin{aligned}\overleftarrow{m}_0 &= (\mathsf{N}/t)\Big\{1-\Big(1-\frac{1}{\mathsf{N}/t}\Big)^{\overleftarrow{m}_1}\Big\} = (\mathsf{N}/t)\Big\{1-1+\frac{\overleftarrow{m}_1 t}{\mathsf{N}}-\binom{\overleftarrow{m}_1}{2}\Big(\frac{t}{\mathsf{N}}\Big)^2+\cdots\Big\}\\ &= \overleftarrow{m}_1 + O\Big(\frac{(\overleftarrow{m}_1)^2 t}{\mathsf{N}}\Big) = \overleftarrow{m}_1\Big\{1+O\Big(\frac{1}{t}\Big)\Big\}.\end{aligned}$$

Thus, we may approximate $\overleftarrow{m}_0$ with $\overleftarrow{m}_1$, unless $t$ is very small. □

More generally, it is possible to show $\overleftarrow{m}_{i+1} \approx \overleftarrow{m}_i$, but it would be unwise to iteratively combine these approximations too many times to conclude $\overleftarrow{m}_j \approx \overleftarrow{m}_i$, for every $j$ and $i$. In fact, it is easy to argue as in [10] that

$$\overleftarrow{m}_k = |\mathtt{DM}|\Big(1-\frac{1}{t}\Big)^{k-1}\frac{1}{t}, \tag{6}$$

for $k \geq 1$, so that $\overleftarrow{m}_j = \big(1-\frac{1}{t}\big)^{j-i}\overleftarrow{m}_i$. Here, the $|\mathtt{DM}|$ denotes the number of distinct points expected in a non-perfect DP matrix. To be more precise, the $|\mathtt{DM}|$ used here counts the points that were used as inputs to the iterating function during the non-perfect DP table creation, so that the starting points are included and the ending points are excluded.

In passing, we caution the reader that one must be aware of the possibility of erring when extending (6) to the $k = 0$ case and writing

$$\overleftarrow{m}_0 = |\mathtt{DM}|\Big(1-\frac{1}{t}\Big)^{-1}\frac{1}{t} \approx |\mathtt{DM}|\frac{1}{t}\Big(1+\frac{1}{t}\Big), \qquad \text{(problematic!)} \tag{7}$$

since one can infer from the proof of Lemma 1 that the correct value is closer to

$$\overleftarrow{m}_0 \approx \overleftarrow{m}_1\Big\{1-\frac{(\overleftarrow{m}_1-1)t}{2\mathsf{N}}\Big\} \approx |\mathtt{DM}|\frac{1}{t}\Big(1-\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2t}\Big). \tag{8}$$

These two expressions for $\overleftarrow{m}_0$ are certainly close to each other and also to $\overleftarrow{m}_1$, so that the use of (7) could be acceptable under certain circumstances, but it would be inappropriate to claim (7) by itself.

The core information missing from (6) is also already available. It is known [11] that a single non-perfect DP matrix created with $m_0$ starting points is expected to contain

$$|\mathtt{DM}| = \frac{2m_0 t}{1+\sqrt{1+2\mathsf{D}_{\mathrm{msc}}}} \tag{9}$$

distinct points, where $\mathtt{D}_{\mathrm{msc}} = \frac{m_0 t^2}{\mathsf{N}}$ is the matrix stopping constant for the non-perfect DP matrix. The information we have gathered so far can be used to related the number of starting points to the number of distinct ending points.

**Lemma 2** *A non-perfect DP matrix created with $m_0$ starting points is expected to contain $\frac{2m_0}{1+\sqrt{1+2\mathtt{D}_{\mathrm{msc}}}}$ non-merging chains, where $\mathtt{D}_{\mathrm{msc}} = \frac{m_0 t^2}{\mathsf{N}}$. Conversely, to create a perfect DP matrix containing $m$ non-merging chains, one must expect to generate $m_0 = \left(1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\right)m$ chains.*

*Proof* Lemma 1 and (6) together imply that $\frac{|\mathtt{DM}|}{t}$ is the number of non-merging chains. Hence, the first claim follows from (9).

As for the second claim, it suffices to solve for $m_0$ in

$$m = \frac{|\mathtt{DM}|}{t} = \frac{2m_0}{1+\sqrt{1+2m_0 t^2/\mathsf{N}}}.$$

After rewriting this in the form

$$1 + \sqrt{1 + 2\bar{\mathtt{D}}_{\mathrm{msc}}\frac{m_0}{m}} = 2\,\frac{m_0}{m},$$

one can solve for $\frac{m_0}{m}$, so as to express $m_0$ as a function of $\bar{\mathtt{D}}_{\mathrm{msc}}$ multiplied by $m$. □

Note that the first sentence of this lemma gives a simple formula for the number of non-merging chains $\alpha$, discussed in Section 2.2, which many previous works had attempted to find.

For the remainder of this section,

$$m_0 = \left(1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\right)m \tag{10}$$

will always denote the number of starting points that are required to create a perfect DP table containing $m$ non-merging chains. This equation is equivalent to

$$\mathtt{D}_{\mathrm{msc}} = \left(1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\right)\bar{\mathtt{D}}_{\mathrm{msc}}, \tag{11}$$

and again to

$$\bar{\mathtt{D}}_{\mathrm{msc}} = \sqrt{1 + 2\mathtt{D}_{\mathrm{msc}}} - 1, \tag{12}$$

which can be used to convert any formula given in terms of $\bar{\mathtt{D}}_{\mathrm{msc}}$ into one given in terms of $\mathtt{D}_{\mathrm{msc}}$.

Another interesting formula that follows from the notational convention (10) is

$$|\mathtt{DM}| = mt, \tag{13}$$

which is evident from the first equation in the proof to Lemma 2. That is, the non-perfect DP matrix created from $m_0$ starting points, as given by (10), is expected to cover $mt$ distinct points.

The pre-computation phase of a perfect DP tradeoff requires $m_0 t\ell$ iterations of the one-way function. We define the *pre-computation coefficient* for the perfect DP tradeoff to be $\bar{\mathtt{D}}_{\mathrm{pc}} = \frac{m_0 t\ell}{\mathsf{N}}$, so that the cost of pre-computation is $\bar{\mathtt{D}}_{\mathrm{pc}}\mathsf{N}$. The following statement is a direct consequence of Lemma 2.

**Proposition 3** *The pre-computation coefficient of the perfect DP tradeoff is*

$$\bar{\mathtt{D}}_{\mathrm{pc}} = \Big(1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\Big)\frac{mt\ell}{\mathsf{N}}.$$

By the definition of the coverage rate, the success probability of the perfect DP tradeoff may be stated as

$$\bar{\mathtt{D}}_{\mathrm{ps}} = 1 - \Big(1 - \frac{mt\,\bar{\mathtt{D}}_{\mathrm{cr}}}{\mathsf{N}}\Big)^{\ell} = 1 - \exp\Big(-\frac{mt\ell}{\mathsf{N}}\bar{\mathtt{D}}_{\mathrm{cr}}\Big), \tag{14}$$

and we can combine this with Proposition 3 to claim the following.

**Proposition 4** *The success probability of the perfect DP tradeoff is*

$$\bar{\mathtt{D}}_{\mathrm{ps}} = 1 - \exp\Big(-\frac{2\,\bar{\mathtt{D}}_{\mathrm{pc}}\,\bar{\mathtt{D}}_{\mathrm{cr}}}{2 + \bar{\mathtt{D}}_{\mathrm{msc}}}\Big).$$

We have succeeded in obtaining expressions for $\bar{\mathtt{D}}_{\mathrm{pc}}$ and $\bar{\mathtt{D}}_{\mathrm{ps}}$ that does not involve $m_0$. Our next short term objective is to obtain such an expression for $\bar{\mathtt{D}}_{\mathrm{cr}}$. Some technical lemmas need to be prepared first.

Given a function $F : \mathscr{N} \to \mathscr{N}$ and a nonnegative integer $k$, we define $\mathscr{D}_k(F)$ or $\mathscr{D}_k$ to be the set of elements of $\mathscr{N}$ that are $k$-many $F$-iterations away from their closest DPs. In particular, $\mathscr{D}_0$ is the set of DPs. It is clear that $\{\mathscr{D}_k(F)\}_{k=0}^{\infty}$ is a partition of $\mathscr{N}$, and that we can expect the sizes of these subsets to be

$$|\mathscr{D}_k| = \mathsf{N}\Big(1 - \frac{1}{t}\Big)^k \frac{1}{t}, \tag{15}$$

for a random function.

**Lemma 5** *Let $F : \mathscr{N} \to \mathscr{N}$ be chosen uniformly at random from the set of all functions acting on $\mathscr{N}$ and let us fix a set $D \subset \mathscr{D}_k(F)$ for some $k \geq 1$. Then the expect sizes of its iterated images under $F$ will satisfy*

$$\frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}} \;=\; 1 - \exp\left(-\frac{|F^{i-1}(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\right),$$

*for each $i = 1, \ldots, k$.*

*Proof* For a random function $F : \mathscr{A} \to \mathscr{B}$ defined on finite sets and a subset $\mathscr{C}$ of the domain $\mathscr{A}$, the image size is expected to be

$$|F(\mathscr{C})| = |\mathscr{B}|\Big\{1 - \Big(1 - \frac{1}{|\mathscr{B}|}\Big)^{|\mathscr{C}|}\Big\} = |\mathscr{B}|\Big\{1 - \exp\Big(-\frac{|\mathscr{C}|}{|\mathscr{B}|}\Big)\Big\}.$$

The claim is now a direct consequence of the set sizes given by (15). A more detailed proof is provided in Appendix B, for those interested in the subtleties hidden behind this short argument. □

It is possible to work out the iterations expressed by this lemma and write down each iterated image size as a closed-form formula.

**Lemma 6** *Let $F : \mathscr{N} \to \mathscr{N}$ be a random function and let $D \subset \mathscr{D}_k(F)$, for some $k \geq 0$. When $|D| = O(m)$, the size of the i-th iterated image under $F$ is expected to be*

$$|F^i(D)| = \frac{2|D|}{2 + \bar{\mathtt{D}}_{\mathrm{msc}} \frac{|D|}{m} e^{\frac{k}{t}} (1 - e^{-\frac{i}{t}})},$$

*for each $0 \leq i \leq k$.*

*Proof* Let us temporarily introduce the notation $f_i = \frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}$, and rewrite Lemma 5 as

$$f_i = 1 - \exp\Big\{-\Big(1 - \frac{1}{t}\Big) f_{i-1}\Big\} = \Big(1 - \frac{1}{t}\Big) f_{i-1} - \frac{1}{2}\Big(1 - \frac{1}{t}\Big)^2 f_{i-1}^2 + \cdots.$$

The condition $|D| = O(m)$ implies $f_i = O\big(\frac{1}{t}\big)$ so that we can state

$$f_i - f_{i-1} = -\frac{1}{t} f_{i-1} - \frac{1}{2} f_{i-1}^2 + O\Big(\frac{f_{i-1}^2}{t}\Big).$$

Noting that $\frac{f_{i-1}^2}{t}$ is of strictly smaller order than $\frac{f_{i-1}}{t} + \frac{f_{i-1}^2}{2}$, one solves the corresponding differential equation

$$f'(x) = -\frac{1}{t} f(x) - \frac{1}{2} f(x)^2$$

with the initial condition $f(0) = f_0 = \frac{|D|t}{\mathsf{N}e^{-\frac{k}{t}}}$, to obtain

$$f_i = \frac{2|D|t}{2\mathsf{N}e^{\frac{i-k}{t}} + (e^{\frac{i}{t}} - 1)|D|t^2},$$

as an application of the Euler method. It now suffices to combine this with the definition of $f_i$ and approximate appropriately to arrive at the claim. □

The previous two lemmas were prepared to support the next lemma, which gives the probability for a single chain to merge into a set of chains. This information will be used to learn about how a perfect DP table is formed from a non-perfect DP table and also to study the occurrences of false alarms.

**Lemma 7** *Let $F : \mathscr{N} \to \mathscr{N}$ be a random function and let $D \subset \mathscr{D}_k(F)$, for some k. When $|D| = O(m)$, the probability for a random point $x \in \mathscr{D}_k(F)$ to satisfy $F^k(x) \notin F^k(D)$ is*

$$\Big\{1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2} \frac{|D|}{m} \big(e^{\frac{k}{t}} - 1\big)\Big\}^{-2}.$$

*Proof* The probability in question is given by

$$\begin{aligned}\prod_{i=0}^{k} \Big(1 - \frac{|F^i(D)|}{|\mathscr{D}_{k-i}|}\Big) &= \prod_{i=0}^{k} \Big(1 - \frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\Big) \\ &= \Big(1 - \frac{|D|t}{\mathsf{N}e^{-\frac{k}{t}}}\Big) \prod_{i=1}^{k} \Big(1 - \frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\Big).\end{aligned}$$

By applying Lemma 5 to the product of $k$ terms, we can write

$$\prod_{i=1}^{k}\Big(1-\frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\Big)=\prod_{i=1}^{k}\exp\Big(-\frac{|F^{i-1}(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\Big)$$
$$=\exp\left(-\Big(1-\frac{1}{t}\Big)\sum_{i=0}^{k-1}\frac{|F^i(D)|}{\mathsf{N}(1-\frac{1}{t})^{k-i}\frac{1}{t}}\right).$$

Since we are given the condition $|D|=O(m)$, we can apply Lemma 6, or the last equation in its proof, and compute the sum inside the exponential function as

$$\sum_{i=0}^{k-1}\frac{|F^i(D)|}{\mathsf{N}\big(1-\frac{1}{t}\big)^{k-i}\frac{1}{t}}=\sum_{i=0}^{k-1}\frac{2|D|t}{2\mathsf{N}e^{\frac{i-k}{t}}+(e^{\frac{i}{t}}-1)|D|t^2}$$
$$=\int_0^{k/t}\frac{2|D|t}{\frac{2\mathsf{N}}{t}e^{-\frac{k}{t}}e^u+(e^u-1)|D|t}\,du=2\ln\Big\{1+\frac{|D|t^2}{2\mathsf{N}}\big(e^{\frac{k}{t}}-1\big)\Big\}.$$

By substituting the sum back into the exponential function, we get

$$\prod_{i=0}^{k}\Big(1-\frac{|F^i(D)|}{|\mathscr{D}_{k-i}|}\Big)=\Big(1-\frac{|D|t}{\mathsf{N}e^{-\frac{k}{t}}}\Big)\Big\{1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\frac{|D|}{m}\big(e^{\frac{k}{t}}-1\big)\Big\}^{-2(1-\frac{1}{t})}.$$

The $\big(1-\frac{1}{t}\big)$ term in the exponent is insignificant and the condition $|D|=O(m)$ allows us to ignore the first product term. Hence, we arrive at the claimed formula. □

With the help of the technical lemmas that have been prepared, we can finally present something of more direct practical value.

**Proposition 8** *The coverage rate of a perfect DP matrix is*

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

*Proof* Consider a pre-computed non-perfect DP matrix and the process of removing chains to obtain a perfect DP matrix. A chain survives through the collision removal process if and only if it does not collide with another chain that is longer than (or equal to) its length. Hence, according to Lemma 7, the probability for a chain of length $k$ in a non-perfect DP table to remain in the perfect table is

$$\Big\{1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\frac{\overleftarrow{m}_k}{m}\big(e^{\frac{k}{t}}-1\big)\Big\}^{-2}.$$

This figure is a slight underestimate since the collisions among chains of the same length were reflected too many times, but such collisions are rare and will not cause noticeable inaccuracy.

Since the number of, possibly merging, chains of length $k$ is $m_0\big(1-\frac{1}{t}\big)^{k-1}\frac{1}{t}$ and the perfect table contains no overlapping of points, the number of distinct points in the perfect DP table is

$$\sum_{k=1}^{\infty}k\cdot m_0\Big(1-\frac{1}{t}\Big)^{k-1}\frac{1}{t}\cdot\Big\{1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\frac{\overleftarrow{m}_k}{m}\big(e^{\frac{k}{t}}-1\big)\Big\}^{-2}.$$

This formula does not count the ending points and only includes the points that were used as inputs to the iterating function during the DP table computation.

The coverage rate of the perfect DP matrix is given by

$$\bar{\mathtt{D}}_{\mathrm{cr}} = \frac{1}{mt} m_0 \Big(1-\frac{1}{t}\Big)^{-1} \sum_{k=1}^{\infty} \frac{k}{t} \cdot e^{-\frac{k}{t}} \cdot \Big\{1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2} e^{-\frac{k}{t}} \Big(1-\frac{1}{t}\Big)^{-1} \big(e^{\frac{k}{t}}-1\big)\Big\}^{-2},$$

where we have used (6) and (13) to remove the $\overleftarrow{m}_k$ term. After ignoring the insignificant $(1-\frac{1}{t})^{-1}$ terms, the coverage rate can be computed as

$$\bar{\mathtt{D}}_{\mathrm{cr}} = \frac{m_0}{m} \int_0^{\infty} u e^{-u} \Big\{1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\big(1-e^{-u}\big)\Big\}^{-2} du = \frac{m_0}{m} \frac{\ln\big(1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\big)}{\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\big(1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\big)}.$$

It now suffices to recall Lemma 2 to arrive at the claimed formula. □

Let us briefly digress and recall the average chain length $\bar{\beta}$ of a perfect DP matrix, introduced in Section 2.2. By definition, it is the number of points in a perfect DP matrix divided by the number of its ending points, and we can easily write it as

$$\bar{\beta} = \frac{|\bar{\mathtt{DM}}|}{m} = \frac{mt\bar{\mathtt{D}}_{\mathrm{cr}}}{m} = t\,\frac{2}{\bar{\mathtt{D}}_{\mathrm{msc}}} \ln\Big(1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\Big). \tag{16}$$

Should it be required, we can use (12) to rewrite this in terms of the parameters $m_0$ and $t$ as

$$\bar{\beta} = t\,\frac{1+\sqrt{1+2\mathtt{D}_{\mathrm{msc}}}}{\mathtt{D}_{\mathrm{msc}}} \ln\Big(\frac{1+\sqrt{1+2\mathtt{D}_{\mathrm{msc}}}}{2}\Big), \tag{17}$$

where $\mathtt{D}_{\mathrm{msc}} = \frac{m_0 t^2}{\mathtt{N}}$. It is easy to check that this $\bar{\beta}$ value is always smaller than the average chain length $\beta_0 = t$ before the removal of merging chains. Even though we are keeping the longest of any set of merging chains, the longer chains are more likely to merge into one another and be discarded.

The technical statement of Lemma 7 is also the key to obtaining the cost associated with alarms.

**Proposition 9** *The number of one-way function invocations required to resolve alarms while processing a single perfect DP table is expected to be*

$$2t\,\mathrm{Li}_2\Big(\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2+\bar{\mathtt{D}}_{\mathrm{msc}}}\Big),$$

*where* $\mathrm{Li}_2(z) = \sum_{n=1}^{\infty} \frac{z^n}{n^2} = -\int_0^z \frac{\ln(1-u)}{u}\,du$ *is the dilogarithm function.*

*Proof* Pre-computation chains are generated well before any online chain is computed, but let us first fix an online chain and treat the pre-computation chains as if they were freshly generated after the creation of the online chain.

Suppose we have an online chain of length $i$ that has terminated at a DP. Let us compute the expected work required to deal with the alarm which may or may not result from this DP chain. We treat each pre-computation chain as if it were being generated afresh and study how it might collide with the current online chain.

Let us compute the probability for the randomly generated pre-computation chain to collide with the given online chain of length $i$ at the $j$-th iteration of the pre-computation chain and become a DP chain of length $k$. The pre-computation chain iterations must follow the online chain iterations from the point of chain merge and the probability we seek is nonzero if and only if $j \le k \le j+i$. For a $k$ that falls within this range, the starting point and

the first $(j-1)$ iterations of the pre-computation chain must be chosen among the non-DPs that do not belong to the length-$i$ online chain and the $j$-th iteration must land on the unique online chain point that results in the chain becoming a DP chain of length $k$. Hence, the probability for a merge at the $j$-th iteration that leads to a length-$k$ DP chain is given by

$$\left(1-\frac{1}{t}-\frac{i}{\mathsf{N}}\right)^{j}\frac{1}{\mathsf{N}}\approx e^{-\frac{j}{t}}\frac{1}{\mathsf{N}}.$$

The approximation is valid since $\frac{i}{\mathsf{N}}$ is of much smaller order than $\frac{1}{t}$.

The collision discussed above will result in an alarm during the online phase if and only if the freshly generated pre-computation chain remains in the perfect DP matrix even after the treatment of merging pre-computation chains. As was discussed during the proof of Proposition 8, we can combine Lemma 7 and (6) to write the probability for the pre-computation chain to escape the chain removal process as

$$\left\{1+\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2}\left(1-e^{-\frac{k}{t}}\right)\right\}^{-2}.$$

Since a complete record of the online chain is maintained, regeneration of the pre-computation chain to resolve an alarm that may result from the merge discussed above can be stopped at the $j$-th iteration. Combining the contribution of all $m_0$ pre-computation chains, the expected cost of resolving alarms associated with a single perfect DP table, which may or may not occur from a single online DP chain of length $i$ can be written as

$$\sum_{j=1}^{\infty}\sum_{j\le k\le j+i} j\cdot e^{-\frac{j}{t}}\frac{1}{\mathsf{N}}\cdot m_0\cdot\left\{1+\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2}\left(1-e^{-\frac{k}{t}}\right)\right\}^{-2}.$$

Finally, considering the probability for an online chain to become a DP chain of length $i$, the number of function iterations expected from the processing of a single perfect DP table can be written as

$$\begin{aligned}
&\sum_{i=1}^{\infty}e^{-\frac{i}{t}}\frac{1}{t}\sum_{j=1}^{\infty}\sum_{j\le k\le j+i} j\cdot e^{-\frac{j}{t}}\frac{1}{\mathsf{N}}\cdot m_0\cdot\left\{1+\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2}\left(1-e^{-\frac{k}{t}}\right)\right\}^{-2}\\
&\quad=\frac{m_0t^3}{\mathsf{N}}\int_0^{\infty}\int_0^{\infty}\int_y^{y+x}e^{-x}ye^{-y}\left\{1+\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2}\left(1-e^{-z}\right)\right\}^{-2}dz\,dy\,dx\\
&\quad=t\,\bar{\mathsf{D}}_{\mathrm{msc}}\frac{m_0}{m}\frac{4}{\bar{\mathsf{D}}_{\mathrm{msc}}(2+\bar{\mathsf{D}}_{\mathrm{msc}})}\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right).
\end{aligned}$$

The claim is reached by combining this with Lemma 2. □

Having obtained the cost of dealing with alarms, the online complexities of the perfect DP tradeoff can be expressed as a single tradeoff curve.

**Theorem 10** *The time memory tradeoff curve for the perfect DP tradeoff is $TM^2=\bar{\mathsf{D}}_{\mathrm{tc}}\mathsf{N}^2$, where the tradeoff coefficient is given by*

$$\bar{\mathsf{D}}_{\mathrm{tc}}=\left\{1+2\,\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right)\right\}\frac{\bar{\mathsf{D}}_{\mathrm{ps}}\left\{\ln(1-\bar{\mathsf{D}}_{\mathrm{ps}})\right\}^2}{\bar{\mathsf{D}}_{\mathrm{msc}}\,\bar{\mathsf{D}}_{\mathrm{cr}}^3}.$$

*Here,* $\mathrm{Li}_2(z)=\sum_{n=1}^{\infty}\frac{z^n}{n^2}=-\int_0^z\frac{\ln(1-u)}{u}\,du$ *is the dilogarithm function.*

*Proof* The probability for the $i$-th DP table to be processed during the online phase executed for a single inversion target is $\left(1-\frac{mt\bar{\mathsf{D}}_{\mathrm{cr}}}{\mathsf{N}}\right)^{i-1}$. Online processing of each table is expected to requires $t$ iterations of the one-way function for the online chain creation and the number of iterations required to deal with alarms is given by Proposition 9. Hence, the number of one-way function iterations expected during the online phase is

$$T=\sum_{i=1}^{\ell}\left(1-\frac{mt\bar{\mathsf{D}}_{\mathrm{cr}}}{\mathsf{N}}\right)^{i-1}\left\{1+2\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right)\right\}t=\frac{\bar{\mathsf{D}}_{\mathrm{ps}}}{\bar{\mathsf{D}}_{\mathrm{msc}}\bar{\mathsf{D}}_{\mathrm{cr}}}\left\{1+2\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right)\right\}t^2,$$

where the second equality relies on Proposition 4. The tradeoff curve is obtained by combining this time complexity $T$ with the storage complexity $M=m\ell$ as follows:

$$\begin{aligned}TM^2&=\frac{\bar{\mathsf{D}}_{\mathrm{ps}}}{\bar{\mathsf{D}}_{\mathrm{msc}}\bar{\mathsf{D}}_{\mathrm{cr}}}\left\{1+2\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right)\right\}(mt\ell)^2\\&=\frac{\bar{\mathsf{D}}_{\mathrm{ps}}}{\bar{\mathsf{D}}_{\mathrm{msc}}\bar{\mathsf{D}}_{\mathrm{cr}}}\left\{1+2\mathrm{Li}_2\left(\frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2+\bar{\mathsf{D}}_{\mathrm{msc}}}\right)\right\}\left\{\frac{\ln(1-\bar{\mathsf{D}}_{\mathrm{ps}})}{\bar{\mathsf{D}}_{\mathrm{cr}}}\right\}^2\mathsf{N}^2.\end{aligned}$$

Once again, Proposition 4 is required to obtain the second equality. □

The appearance of the less familiar dilogarithm function should not be associated with reduced practical usability of the above formula. The dilogarithm function can be handled by numeric computations softwares as easily as any other more frequently encountered functions such as log, exp, or the trigonometric functions, and its algebraic manipulations, including differentiation and indefinite integration, are also easy.

### 3.2 Storage Optimization

An analysis of the perfect DP tradeoff would not be complete without a discussion of the storage optimization techniques. Dealing with the starting point storage is quite straightforward. One requires $\log m_0$ bits of space for every starting point, and (10) implies that this will be one or two bits more than $\log m$ for parameters of interest. Hence, one may safely claim that the number of bits required to store a single starting point for a perfect DP tradeoff is very close to that required for the non-perfect DP tradeoff, when comparable parameters are used by the two algorithms.

To deal with the ending point storage, one needs to discuss the effects of truncating ending points before storage. Consider an ending point truncation method for which two random DPs, truncated in the specified manner, will have probability $\frac{1}{r}$ of matching with each other. We shall express such a situation as having a $\frac{1}{r}$ probability of truncated match. One specific way to do this would be to retain just $\log r$ bits of the ending point that are unrelated to the DP definition. Note that we are considering truncations of DPs only and not of the general points of $\mathscr{N}$.

The effects of ending point truncation on the perfect DP tradeoff is slightly different from that on the non-perfect DP tradeoff, which was treated in [11]. The truncation may cause two non-merging pre-computation chains to become indistinguishable at the ending points and cause more chains to be discarded. However, the following lemma shows that these further collisions can mostly be avoided by recording slightly more than $\log m$ bits.

**Lemma 11** *Assume the use of an ending point truncation method with a $\frac{1}{r}$ probability of truncated match. If $m$ distinct DPs are truncated, then we can expect to obtain $r\left\{1-\exp(-\frac{m}{r})\right\}$ distinct truncated points. Conversely, when $r>m$, one must expect to truncate $r\ln\left(\frac{r}{r-m}\right)$ DPs in order to collect $m$ distinct truncated points.*

This lemma is a trivial consequence of treating the truncation process as the random selection of points from a pool of $r$-many points.

Let us consider a specific example. When $r = 2^5 m$ is used for truncation, it suffices to truncate $32m\ln\left(\frac{32}{31}\right) = 1.01596\,m$ DPs in order to obtain $m$ distinct truncated ending points. Combining this information with Proposition 3, one can state that, by recording just $5 + \log m$ bits of each ending point, one can control the extra pre-computation necessitated by the ending point truncation to within approximately 2%. Note that this is not 1.596% and only claimed approximately, because the variable $m$ appears not only in the $\frac{mt\ell}{\mathsf{N}}$ term of Proposition 3, but also inside the $\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}$ term. In any case, the effects of ending point truncation on the collision of ending points can be maintained at an ignorable level by retaining a little more than $\log m$ bits of information through the truncation process. Note that by ignoring the ending point collisions induced by truncations, we are also ignoring their effects on the pre-computation time and also on the coverage rate, or, equivalently, the success probability.

We now need to discuss the effects of truncation on the online time. The terminating DP of the online chain must be searched for among the truncated ending points, so we have the possibility of falsely announcing a match and then regenerating the pre-computation chain to resolve this alarm.

**Lemma 12** *Assume the use of an ending point truncation method with a $\frac{1}{r}$ probability of truncated match. Further assume that $r$ has been chosen to be large enough so that the occurrences of indistinguishable ending points caused by truncations are sufficiently limited to be ignored. Then the number of extra one-way function invocations induced by truncation-related alarms is expected to be*

$$t\,\frac{m}{r}\,\frac{2}{\bar{\mathtt{D}}_{\mathrm{msc}}(1+\bar{\mathtt{D}}_{\mathrm{msc}})}\ln\left(1+\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2}\right),$$

*for each fully processed perfect DP table.*

*Proof* Let us compute the probability for an online chain to become a DP chain of length $i$ and not merge into the perfect DP matrix, but have a truncated ending point that coincides with a truncated ending point in the perfect DP table. For this event to occur, the online chain must be created in the following manner: (1) Random choices for the first $i$ nodes of the online chain, starting from the correct pre-image of the inversion target, must be made among the non-DPs that do not belonging to the non-perfect DP matrix, which is seen before the removal of merging chains; (2) The final point is chosen among DPs that is different from the $m$ ending points; (3) Furthermore, the final point must be chosen so that its truncation matches one of the $m$ truncated ending points. The process (2) and (3) are not quite independent, but since the number of DPs is much greater than the number of points we know the final point not to be, i.e., $\frac{\mathsf{N}}{t} \gg m$, the dependence can be ignored. Thus, the probability we seek is

$$\left(1-\frac{1}{t}-\frac{|\mathtt{DM}|}{\mathsf{N}}\right)^i\left(\frac{1}{t}-\frac{m}{\mathsf{N}}\right)\frac{m}{r} \approx \left(1-\frac{1}{t}-\frac{|\mathtt{DM}|}{\mathsf{N}}\right)^i\frac{1}{t}\frac{m}{r} = \left(1-\frac{1+\bar{\mathtt{D}}_{\mathrm{msc}}}{t}\right)^i\frac{1}{t}\frac{m}{r},$$

where we have used $\frac{m}{\mathsf{N}} = O(\frac{1}{mt}) = o(\frac{1}{t})$ for the approximation and (13) for the final equality. Thus, the probability for the online processing of a perfect DP table to cause a truncation-related alarm, i.e., an alarm that does not involve the online chain merging into the pre-computation matrix, is given by

$$\sum_{i=1}^{\infty}\left(1-\frac{1+\bar{\mathtt{D}}_{\mathrm{msc}}}{t}\right)^i\frac{1}{t}\frac{m}{r} = \frac{1-\frac{1+\bar{\mathtt{D}}_{\mathrm{msc}}}{t}}{\frac{1+\bar{\mathtt{D}}_{\mathrm{msc}}}{t}}\frac{1}{t}\frac{m}{r} \approx \frac{1}{1+\bar{\mathtt{D}}_{\mathrm{msc}}}\frac{m}{r}.$$

Notice that the length of a pre-computation chain is independent of how likely it is to be involved in a truncation-related alarm. Hence, the number of iterations required to regenerated the pre-computation chain involved with such a pseudo-collision is expected to be the average chain length of the perfect DP matrix, which is given by (16). The cost of resolving alarms that are induced by truncation is

$$\frac{1}{1+\bar{\mathrm{D}}_{\mathrm{msc}}}\frac{m}{r}t\frac{2}{\bar{\mathrm{D}}_{\mathrm{msc}}}\ln\Big(1+\frac{\bar{\mathrm{D}}_{\mathrm{msc}}}{2}\Big),$$

for the full processing of a single perfect DP table. □

The normal one-way function iterations required to generate the online chain and deal with alarms while processing a single perfect DP table was stated during the proof of Theorem 10 as

$$t\Big\{1+2\mathrm{Li}_2\Big(\frac{\bar{\mathrm{D}}_{\mathrm{msc}}}{2+\bar{\mathrm{D}}_{\mathrm{msc}}}\Big)\Big\}. \tag{18}$$

If we assume that sufficient information is left after the ending point truncation so that the number of indistinguishable ending points are kept ignorably small, then, at the typical parameter of $\bar{\mathrm{D}}_{\mathrm{msc}}=1$, the expected numbers of normal iterations and truncation-related iterations become $\{1+2\mathrm{Li}_2(\frac{1}{3})\}t=1.73243t$ and $\ln(\frac{3}{2})\frac{m}{r}t=0.405465\frac{m}{r}t$, respectively. For example, at $\log r=5+\log m$, the ending point truncation increases the number of one-way function iterations by a mere $\frac{0.405465\frac{1}{32}t}{1.74243t}\approx 0.73\%$. The following can be stated for the general situation.

**Proposition 13** *Suppose that the online phase of a perfect DP tradeoff implementation that stores each ending point in full requires $T$ iterations of the one-way function to complete. Consider the use of an ending point truncation method with a $\frac{1}{r}$ probability of truncated match, where $\log r=\varepsilon+\log m$. If $\varepsilon$ is large enough for the occurrences of indistinguishable ending points caused by truncations to be ignored, then the implementation with the ending point truncation requires*

$$\frac{2\ln\big(1+\frac{\bar{\mathrm{D}}_{\mathrm{msc}}}{2}\big)}{\bar{\mathrm{D}}_{\mathrm{msc}}(1+\bar{\mathrm{D}}_{\mathrm{msc}})\big\{1+2\mathrm{Li}_2\big(\frac{\bar{\mathrm{D}}_{\mathrm{msc}}}{2+\bar{\mathrm{D}}_{\mathrm{msc}}}\big)\big\}}\frac{T}{2^{\varepsilon}}$$

*additional iterations of the one-way function to complete.*

Let us summarize the situation concerning the storage of each perfect DP table entry. The starting point can be stored using slightly more than $\log m$ bits. Ending point DPs can be truncated so that a little more than $\log m$ bits of information is retained with little effect on the success probability, pre-computation cost, and online time. The index file technique can be used to remove almost $\log m$ further bits per ending point without any loss of information. In conclusion storage of each starting point and ending point pair requires a little more than $\log m$ bits. This was also the conclusion obtained for the non-perfect DP tradeoff in [11].

### 3.3 Experimental Results

We have verified the correctness of major parts of our complexity analysis with experiments. In all our tests, the one-way function was instantiated with the key to ciphertext mapping, under a randomly fixed plaintext, of the blockcipher AES-128. Freshly generated random

**Table 1** The number of DP chains before and after removal of chain merges and the coverage rate of the perfect DP matrix. ($\mathsf{N} = 2^{40}$; $\hat{t} = 15t$).

| $m$ | $t$ | $\bar{\mathsf{D}}_{\mathrm{msc}}$ | $m_0$ used | test $m$ | theoretic $\bar{\mathsf{D}}_{\mathrm{cr}}$ | test $\bar{\mathsf{D}}_{\mathrm{cr}}$ |
|---|---|---|---|---|---|---|
| 2000 | $2^{14}$ | 0.48828 | 2488 | 2000.88 | 0.89475 | 0.89302 |
| 4000 | $2^{14}$ | 0.97656 | 5953 | 3996.01 | 0.81433 | 0.81412 |
| 6000 | $2^{14}$ | 1.46484 | 10394 | 5996.79 | 0.75028 | 0.74934 |
| 10000 | $2^{13}$ | 0.61035 | 13051 | 10005.45 | 0.87274 | 0.87319 |
| 20000 | $2^{13}$ | 1.22070 | 32207 | 20001.52 | 0.78062 | 0.78079 |
| 30000 | $2^{13}$ | 1.83105 | 57465 | 30003.72 | 0.70997 | 0.71020 |

plaintexts were used to create different one-way functions that were required for repetitions of the same test. Bit-masking of ciphertexts to 40 bits and its zero-extension to 128-bit keys were used to restrict the search space to a manageable size of $\mathsf{N} = 2^{40}$.

Our first experiment was designed to verify Lemma 2 and Proposition 8 simultaneously. Lemma 2 relates the number of starting points to the number of non-merging chains in a DP matrix and Proposition 8 presents the coverage rate of the perfect DP matrix.

After fixing suitable parameters $m$ and $t$, we first computed the $m_0$ value, as specified by (10). We generated chains from $m_0$ distinct starting points and recorded their terminating DPs, together with their respective chain lengths. A small number of chains that extended beyond the moderately large chain length bound of $\hat{t} = 15t$ were discarded during this process. After dealing with chain merges by retaining only the information corresponding to the longest chain among any set of merging chains, the number of remaining DPs were counted. Next, the lengths of the surviving chains were added together and taken as the number of distinct entries in the perfect DP matrix. The obtained count of matrix entries, divided by $mt$, is our test $\bar{\mathsf{D}}_{\mathrm{cr}}$ value. The whole process was repeated 200 times for each choice of parameter set and the obtained values were averaged.

The test results are summarized in Table 1, together with the integer $m_0$ values we have used and the theoretically computed coverage rates. In each row, the reported number of distinct ending points that resulted from our theoretically computed $m_0$ starting points is very close to the targeted $m$ value, in spite of the small number of test repetitions. It can also be seen that our theory was able to predict the coverage rates accurately.

Even though this test gives some confidence as to the correctness of our theory, let us present another test that makes sure that our accurate predictions of the coverage rate did not result from some lucky averaging effect that conveniently hid logical errors in our lower level arguments.

Recall that the proofs of Proposition 8 and Proposition 9 relied heavily on our ability to write the probability for a random chain of length $k$ not to merge into any of the chains in a non-perfect DP matrix that are longer than $k$. More specifically, this probability was taken to be

$$\left\{1 + \frac{\bar{\mathsf{D}}_{\mathrm{msc}}}{2}\left(1 - e^{-\frac{k}{t}}\right)\right\}^{-2} \tag{19}$$

in both proofs, and was interpreted as the probability for a chain in a non-perfect DP matrix to survive through the process of removing chain merges.

To test this core logic, we first generated multiple non-perfect DP matrices, discarding the small number of chains reaching the length bound of $\hat{t} = 15t$. Then, for each $1 \leq k < \hat{t}$, we counted and recorded the total number of chains of length $k$ found among these matrices. Next, we removed merges from each of the DP matrices to create multiple perfect DP matrices and, once again, recorded the number of chains of each length. We took the ratio

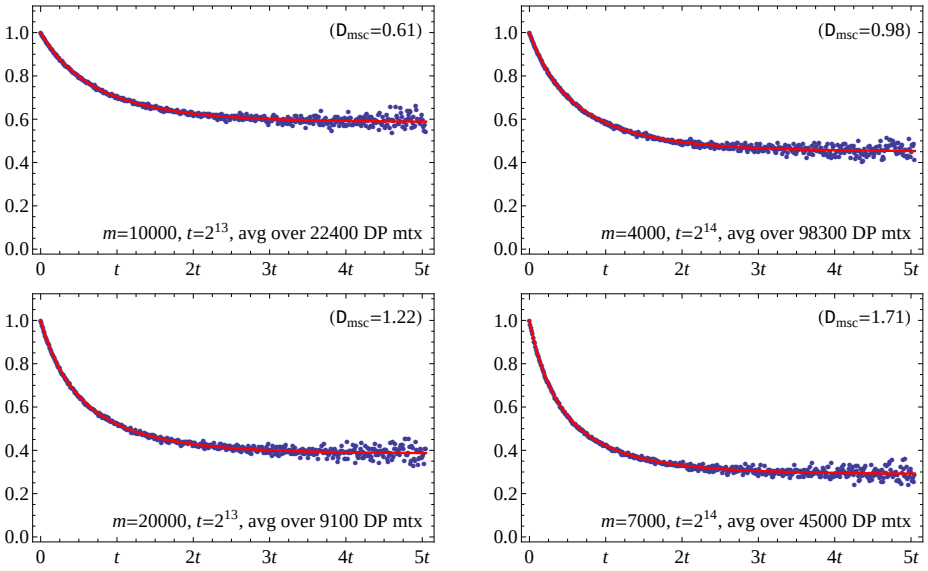


**Fig. 1** The probability ($y$-axis) for DP chains of each length ($x$-axis) to survive through the treatment of merging chains in a DP matrix. (test: *dots*; theory: *line*; $\mathsf{N} = 2^{40}$; $\hat{t} = 15t$).

of the two chain counts, for each length $k$, as our test value of the probability for chains of length $k$ to survive through the chain merge removal process. Note that this ratio of counts cannot be computed separately for each DP matrix and then average later over multiple DP matrices, since the number of chains of any given length is likely to be very small and often zero for any single DP matrix.

The test results are provided in Figure 1. The probability ($y$-axis) for chain survival through the chain merge removal process is given for each chain length ($x$-axis). The lines correspond to our theory, as given by (19), and the dots present the count ratios obtained through tests. Even though our chain length bound was $\hat{t} = 15t$, we have displayed the data only for chain lengths less than approximately $5t$. Furthermore, in each box, we only plotted approximately 500 dots that are equally spaced in terms of chain length values, since densely packing all $5t$ dots into each box made the graphs harder to comprehend.

Our theory matches the experimental data well in all the boxes. Notice that the test results are less reliable at the large chain lengths. This is because longer DP chains appear less frequently and these large chain length data were obtained from a smaller number of chains. A much larger number of DP matrices would need to be generated to obtain meaningful test values at lengths much larger than $5t$.

## 4 Perfect Rainbow Tradeoff

This section gathers facts concerning the perfect rainbow tradeoff that are required for our later comparison of tradeoff algorithms. Even though much of the material given here have not appeared before in the form presented here, the technical core of our complexity analyses were developed by previous works, and the arguments and proofs of this section contain no new ideas. Given enough time, anyone with a full understanding of the works [12], [9], and [11] should be able to reproduce the claims of this section.

### 4.1 Online Efficiency

Unlike the perfect DP tradeoff case, the difficult parts of the complexity analysis for the perfect rainbow tradeoff have already been done by previous works, and it only remains to combine these in an appropriate way.

**Lemma 14** *A non-perfect rainbow matrix created with $m_0$ starting points is expected to contain $\frac{2m_0}{2+\mathtt{R}_{\mathrm{msc}}}$ non-merging chains, where $\mathtt{R}_{\mathrm{msc}} = \frac{m_0 t}{\mathsf{N}}$. Conversely, to create a perfect rainbow matrix that contains m non-merging chains, one must expect to generate $m_0 = \frac{2}{2-\bar{\mathtt{R}}_{\mathrm{msc}}}\, m$ chains.*

*Proof* Consider a non-perfect rainbow matrix created with $m_0$ starting points. It is known [1, 9] that number of distinct points $m_i$ expected in the $i$-th column of this matrix, satisfies

$$\frac{m_i}{\mathsf{N}} \approx \frac{1}{\frac{\mathsf{N}}{m_0} + \frac{i}{2}},$$

for each $0 \le i \le t$. Setting $i = t$ gives the number of distinct ending points $m_t$, which is the first claim of this lemma. To obtain the second claim, it suffices to solve for $m_0$ after substituting $i = t$ and $m_t = m$. □

For the remainder of this section,

$$m_0 = \frac{2}{2-\bar{\mathtt{R}}_{\mathrm{msc}}}\, m \tag{20}$$

will always denote the number of starting points that are required to create a perfect rainbow table containing $m$ non-merging chains.

An interesting situation is when $m_0 = \mathsf{N}$, which is commonly referred to as the maximal perfect rainbow tradeoff [1]. Since a larger number of starting points $m_0$ brings about a larger number of non-merging ending point $m$, this is also the case for which $\bar{\mathtt{R}}_{\mathrm{msc}} = \frac{mt}{\mathsf{N}}$ is at its largest. With an easy manipulation of (20) after substitution of $m_0 = \mathsf{N}$, one finds that there is an upper bound

$$\bar{\mathtt{R}}_{\mathrm{msc}} \le \frac{2t}{t+2} < 2 \tag{21}$$

on the possible range of $\bar{\mathtt{R}}_{\mathrm{msc}}$.

The pre-computation phase of a perfect rainbow tradeoff requires $m_0 t \ell$ iterations of the one-way function. As in the DP case, we define the *pre-computation coefficient* for the perfect rainbow tradeoff to be $\bar{\mathtt{R}}_{\mathrm{pc}} = \frac{m_0 t \ell}{\mathsf{N}}$, so that the number of one-way function iterations required for pre-computation becomes $\bar{\mathtt{R}}_{\mathrm{pc}}\mathsf{N}$. The following statement is a direct consequence of Lemma 14.

**Proposition 15** *The pre-computation coefficient of the perfect rainbow tradeoff is*

$$\bar{\mathtt{R}}_{\mathrm{pc}} = \frac{2}{2-\bar{\mathtt{R}}_{\mathrm{msc}}} \frac{mt\ell}{\mathsf{N}} = \frac{2\bar{\mathtt{R}}_{\mathrm{msc}}}{2-\bar{\mathtt{R}}_{\mathrm{msc}}}\ell.$$

The success probability of a perfect rainbow tradeoff may easily be stated [1] as

$$\bar{\mathtt{R}}_{\mathrm{ps}} = 1 - \Big(1 - \frac{m}{\mathsf{N}}\Big)^{t\ell} = 1 - \exp\Big(-\frac{mt\ell}{\mathsf{N}}\Big) = 1 - \exp\big(-\bar{\mathtt{R}}_{\mathrm{msc}}\ell\big), \tag{22}$$

and this shows that the choice of $\ell$ determines the matrix stopping constant

$$\bar{\mathrm{R}}_{\mathrm{msc}} = -\frac{\ln(1-\bar{\mathrm{R}}_{\mathrm{ps}})}{\ell} \tag{23}$$

one must adhere to, when selecting parameters that achieve a prescribed probability of success. However, one must keep in mind that (21) requires for the number of tables to satisfy

$$\ell > -\frac{1}{2}\ln(1-\bar{\mathrm{R}}_{\mathrm{ps}}). \tag{24}$$

That is, to achieve a given success probability $\bar{\mathrm{R}}_{\mathrm{ps}}$, the number of tables one must use is lower bound by (24). No set of parameters that uses a smaller number of tables can achieve the desired success probability.

Using Proposition 15, we can restate the probability of success (22) as follows.

**Proposition 16** *The success probability of the perfect rainbow tradeoff is*

$$\bar{\mathrm{R}}_{\mathrm{ps}} = 1 - \exp\Big(-\frac{2-\bar{\mathrm{R}}_{\mathrm{msc}}}{2}\,\bar{\mathrm{R}}_{\mathrm{pc}}\Big).$$

Acquiring the online efficiency of the perfect rainbow tradeoff from existing works is also straightforward.

**Theorem 17** *The time memory tradeoff curve for the perfect rainbow tradeoff is* $TM^2 = \bar{\mathrm{R}}_{\mathrm{tc}}\mathsf{N}^2$*, where the tradeoff coefficient is*

$$\begin{aligned}\bar{\mathrm{R}}_{\mathrm{tc}} = &\Big(\bar{\mathrm{R}}_{\mathrm{msc}}\ell - \frac{\bar{\mathrm{R}}_{\mathrm{msc}}}{2} + \ell - 2 + \frac{3}{2\ell}\Big)\\ &- \Big(\frac{\bar{\mathrm{R}}_{\mathrm{msc}}^2\ell}{4} + \bar{\mathrm{R}}_{\mathrm{msc}}\ell^2 - \bar{\mathrm{R}}_{\mathrm{msc}}\ell + \bar{\mathrm{R}}_{\mathrm{msc}} + \ell - 2 + \frac{3}{2\ell}\Big)e^{-\bar{\mathrm{R}}_{\mathrm{msc}}\ell}.\end{aligned}$$

*Proof* According to [9], the expected number of one-way function iterations required to generate the online chain is

$$\ell\Big\{1-(1+\bar{\mathrm{R}}_{\mathrm{msc}}\ell)e^{-\bar{\mathrm{R}}_{\mathrm{msc}}\ell}\Big\}\Big(\frac{t}{\bar{\mathrm{R}}_{\mathrm{msc}}\ell}\Big)^2,$$

and that required to resolve alarms is[3]

$$\Big(\Big\{\bar{\mathrm{R}}_{\mathrm{msc}}\Big(\ell-\frac{1}{2}\Big) - \Big(2-\frac{3}{2\ell}\Big)\Big\} + \Big\{\Big(2-\frac{3}{2\ell}\Big) + \bar{\mathrm{R}}_{\mathrm{msc}}(\ell-1) - \frac{\bar{\mathrm{R}}_{\mathrm{msc}}^2\ell}{4}\Big\}e^{-\bar{\mathrm{R}}_{\mathrm{msc}}\ell}\Big)\Big(\frac{t}{\bar{\mathrm{R}}_{\mathrm{msc}}\ell}\Big)^2.$$

These expected values take the possibility of premature exit from the online phase after discovery of the correct answer into account. The sum of these two terms is the time complexity $T$. We can combine this with the storage complexity $M = m\ell$ and then simplify to arrive at the claim. □

[3] The single $e^{c_{\mathrm{R}}}$ appearing in [9, p.312] should be corrected to $e^{c_{\mathrm{R}}\ell}$.

### 4.2 Storage Optimization

As with the perfect DP tradeoff, storage of a single starting point for the perfect rainbow tradeoff requires $\log m_0$ bits, and (20) shows how this compares with $\log m$. However, unlike the DP case, since $\bar{\mathrm{R}}_{\mathrm{msc}}$ may take values that are very close to 2, there remains the possibility of $\log m_0$ being much larger than $\log m$.

A hint for resolving this problem comes from the derivation process of (21), which shows that $\bar{\mathrm{R}}_{\mathrm{msc}}$ being close to 2 is associated with an unrealistically large amount of pre-computation. In any real-world situation, there will be a bound on the pre-computation cost one is willing to accept. So, let us combine Proposition 15 and (23), and consider a somewhat arbitrary bound of

$$\bar{\mathrm{R}}_{\mathrm{pc}} = \frac{2}{2-\bar{\mathrm{R}}_{\mathrm{msc}}}\big\{-\ln(1-\bar{\mathrm{R}}_{\mathrm{ps}})\big\} \leq 20, \tag{25}$$

on the pre-computation coefficient. Unless the requirement on the success rate is unrealistically small, this will place a reasonably small bound on the coefficient $\frac{2}{2-\bar{\mathrm{R}}_{\mathrm{msc}}}$ of (20), so that $\log m$ will be similar to $\log m_0$. This shows that, for any practical situation, it suffices to allocate slightly more than $\log m$ bits of storage to each starting point.

One side effect of (25) is that it implies the bound $\bar{\mathrm{R}}_{\mathrm{ps}} \leq 1-\frac{1}{e^{20}}$ on the success probability one can consider. However, the appearance of a success probability bound is only natural, since a success probability that is arbitrarily close to 1 cannot be achieved without enormous amount of pre-computation. Furthermore, since $99.999999\% < 1-\frac{1}{e^{20}}$, the implied bound on the success probability is essentially meaningless for even a moderately large bound on the pre-computation cost.

The ending point truncation technique is the subject of our next discussion. In the case of the perfect DP tradeoff, truncation was defined only for the DPs, but ending points may take any form with the rainbow tradeoff, so we now consider truncation of any point from $\mathscr{N}$. Consider a truncation method for which two random points of $\mathscr{N}$, truncated in the specified manner, will have probability $\frac{1}{r}$ of matching with each other. As before, we express such a situation as having $\frac{1}{r}$ probability of truncated match. One specific way to do this would be to truncate to $\log r$ most significant bits.

As in the DP case, truncation may cause two ending points of a perfect rainbow table to become indistinguishable. Concerning this matter, Lemma 11 remains valid for the rainbow tradeoff.

**Lemma 18** *Assume the use of a truncation method with the truncated match probability set to $\frac{1}{r}$. If $m$ distinct ending points of a perfect rainbow matrix are truncated, then we can expect to find $r\big\{1-\exp(-\frac{m}{r})\big\}$ distinct truncated points. Conversely, when $r > m$, one must expect to truncate $r\ln\big(\frac{r}{r-m}\big)$ ending points in order to collect $m$ distinct truncated points.*

The example figure that was given below Lemma 11 remains valid for the perfect rainbow tradeoff. That is, truncation of $1.01596m$ ending points will give $m$ distinct truncated points, when $r = 2^5 m$. Hence, the effects of ending point truncation on pre-computation cost and success probability can be suppressed to an ignorable degree by the use of an $\frac{1}{r}$ such that $\log r = \varepsilon + \log m$ for some small positive integer $\varepsilon$.

Analogous to the DP case, if required, one can work with (20) to find the correct $m_0$ value one must use in order to collect the slightly larger number of non-merging pre-computation chains. Note that our previous discussion of how $\bar{\mathrm{R}}_{\mathrm{msc}}$ is sufficiently bounded away from 2, in practice, implies that the non-linearity hidden within $\bar{\mathrm{R}}_{\mathrm{msc}}$ will not cause too much disturbance. In particular, our claim of each starting point requiring $\log m_0 \approx \log m$ bits of storage

remains valid even if one wants to account for the small loss of pre-computation chains experienced through the truncation of ending points.

The effect of truncation on the online time is considered next.

**Lemma 19** *Assume the use of ending point truncation with the truncated match probability set to $\frac{1}{r}$. Further assume that r has been chosen to be large enough so that the occurrences of indistinguishable ending points from truncations are sufficiently limited to be ignored. Then, during the online phase of the perfect rainbow tradeoff, one can expected to observe*

$$\left(\begin{array}{l}\Big(\bar{\mathsf{R}}_{\mathrm{msc}}\ell^2-\bar{\mathsf{R}}_{\mathrm{msc}}\ell+\dfrac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}-\ell+2-\dfrac{3}{2\ell}\Big)\\ +\Big(\dfrac{\bar{\mathsf{R}}^2_{\mathrm{msc}}\ell}{4}-\bar{\mathsf{R}}_{\mathrm{msc}}\ell+\bar{\mathsf{R}}_{\mathrm{msc}}+\ell-2+\dfrac{3}{2\ell}\Big)e^{-\bar{\mathsf{R}}_{\mathrm{msc}}\ell}\end{array}\right)\frac{m}{r}\Big(\frac{t}{\bar{\mathsf{R}}_{\mathrm{msc}}\ell}\Big)^2$$

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

*Proof* Consider the non-perfect rainbow matrix created with $m_0=\frac{2m}{2-\bar{\mathsf{R}}_{\mathrm{msc}}}$ starting points and let $m_i$ $(0\le i\le t)$ denote the number of distinct points expected in the $i$-th column of this matrix. The probability for the online chain created at the $i$-th iteration, i.e., the online chain of length $i$, starting from the correct inversion target pre-image, not to merge into the perfect rainbow matrix, but have an ending point that truncates to a truncated matrix ending point is

$$\frac{m}{r}\prod_{j=0}^{i}\Big(1-\frac{m_{t-j}}{\mathsf{N}}\Big)\approx\frac{m}{r}\frac{\frac{\mathsf{N}}{m_0}+\frac{t-i-1}{2}}{\frac{\mathsf{N}}{m_0}+\frac{t}{2}}\frac{\frac{\mathsf{N}}{m_0}+\frac{t-i-2}{2}}{\frac{\mathsf{N}}{m_0}+\frac{t-1}{2}}\approx\frac{m}{r}\Big(\frac{\frac{\mathsf{N}}{m_0}+\frac{t-i}{2}}{\frac{\mathsf{N}}{m_0}+\frac{t}{2}}\Big)^2=\frac{m}{r}\Big(1-\frac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}\frac{i}{t}\Big)^2.$$

Here, the first approximation relies on $\frac{m_i}{\mathsf{N}}\approx\big(\frac{\mathsf{N}}{m_0}+\frac{i}{2}\big)^{-1}$, which may be found in [1, 9].

The expected number of extra one-way function iterations induced by truncation-related alarms is given by

$$\ell\sum_{i=0}^{t}(t-i)\frac{m}{r}\Big(1-\frac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}\frac{i}{t}\Big)^2\Big(1-\frac{m}{\mathsf{N}}\Big)^{\ell(i-1)},$$

for the (expected) processing of the $\ell$ rainbow tables. Here, the $\ell(i-1)$-th powered term gives the probability for the $i$-th iteration of the online phase algorithm to be executed. Unless $t$ is very small, this can be approximated by

$$t^2\ell\frac{m}{r}\int_0^1(1-u)\Big(1-\frac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}u\Big)^2\exp\big(-\bar{\mathsf{R}}_{\mathrm{msc}}\ell u\big)\,du.$$

Explicit computation of this definite integral results in our claim. □

Recall that the total online time, without ending point truncation, was given during the proof of Theorem 17. It is straightforward to express the effects of ending point truncation on the total online time.

**Proposition 20** *Suppose that the online phase of a perfect rainbow tradeoff implementation that stores each ending point in full requires T iterations of the one-way function to complete. Consider the use of the ending point truncation with the truncated match probability set to $\frac{1}{r}$, where $\log r=\varepsilon+\log m$. If $\varepsilon$ is large enough for the effects of truncation on pre-computation chain collision to be ignored, then the implementation with the ending point truncation requires*

$$\frac{-\big(\bar{\mathsf{R}}_{\mathrm{msc}}\ell-\frac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}+\ell-2+\frac{3}{2\ell}\big)+\big(\frac{\bar{\mathsf{R}}^2_{\mathrm{msc}}\ell}{4}-\bar{\mathsf{R}}_{\mathrm{msc}}\ell+\bar{\mathsf{R}}_{\mathrm{msc}}+\ell-2+\frac{3}{2\ell}\big)e^{-\bar{\mathsf{R}}_{\mathrm{msc}}\ell}+\bar{\mathsf{R}}_{\mathrm{msc}}\ell^2}{\big(\bar{\mathsf{R}}_{\mathrm{msc}}\ell-\frac{\bar{\mathsf{R}}_{\mathrm{msc}}}{2}+\ell-2+\frac{3}{2\ell}\big)-\big(\frac{\bar{\mathsf{R}}^2_{\mathrm{msc}}\ell}{4}-\bar{\mathsf{R}}_{\mathrm{msc}}\ell+\bar{\mathsf{R}}_{\mathrm{msc}}+\ell-2+\frac{3}{2\ell}\big)e^{-\bar{\mathsf{R}}_{\mathrm{msc}}\ell}-\bar{\mathsf{R}}_{\mathrm{msc}}\ell^2e^{-\bar{\mathsf{R}}_{\mathrm{msc}}\ell}}\frac{T}{2^{\varepsilon}}$$

*additional iterations of the one-way function to complete.*

For the typical parameters of $\bar{\mathtt{R}}_{\mathrm{msc}} = 1$ and $\ell = 1$, the claim is that $0.774568\frac{T}{2^{\varepsilon}}$ additional one-way function iterations are expected due to truncation-related alarms. At $\varepsilon = 5$, this is $0.0242052T$, which implies a 2.42% increase in online time due to ending point truncation.

Let us summarize the issue of storage optimization for the perfect rainbow tradeoff. Each starting point can be stored in slightly more than $\log m$ bits. Each ending point can be truncated to slightly more than $\log m$ bits with little effect on the success probability, pre-computation cost, and online time. The index file technique can be used to remove almost $\log m$ further bits per ending point without any loss of information. In all, storage of each starting point and ending point pair requires a little more than $\log m$ bits. Even though this is identical to the conclusion obtained in [11] for the non-perfect rainbow tradeoff, the analysis had to be repeated here for the perfect rainbow tradeoff.

## 5 Tradeoff Algorithm Performance Comparisons

Formulas for the success rates, online efficiencies, and pre-computation costs of the perfect DP and perfect rainbow tradeoffs were obtained in the previous two sections. This information is used in this section to understand the performances of the two algorithms.

### 5.1 Method of Comparison

Let us briefly summarize the arguments of [11, Section 8.1] that suggest how to compare different tradeoff algorithms.

- Different algorithms should be compared at parameters that achieve the same probability of success.
- The tradeoff coefficient is a measure of the online efficiency of an algorithm. However, one must keep in mind that the unit of storage used while computing the tradeoff curve was the number of entries written to the pre-computation table, rather than the physical amount of required storage space. Hence, one must account for the number of bits required to store each entry, which may be different among algorithms, when comparing the tradeoff coefficients of different algorithms against each other. In the case of the classical Hellman, non-perfect DP, and non-perfect rainbow tradeoffs under typical environments, it is reasonable to compare the *adjusted* tradeoff coefficients $\mathtt{H}_{\mathrm{tc}}$, $\frac{1}{4}\mathtt{D}_{\mathrm{tc}}$, and $\mathtt{R}_{\mathrm{tc}}$ against each other.
- For any one tradeoff algorithm, a parameter set that achieves better online efficiency usually calls for a larger amount of pre-computation. The appropriate balance between these two factors will be different for every user and situation, and the comparison of algorithm performances can only be a subjective matter in this respect. What can be done objectively is to present the pre-computation cost versus online efficiency relation for each algorithm as graphs. The final decision as to which algorithm is most suitable can be made by the user, for the specific situation in hand, based on this information.

We will follow this approach in comparing the perfect DP and perfect rainbow tradeoff algorithms against each other. Recall that the classical Hellman, non-perfect DP, and non-perfect rainbow tradeoffs were compared in [11], and that the rainbow tradeoff could be seen as outperforming the other two in typical situations. All our graphs given below include the

information for the non-perfect rainbow tradeoff, so that the two perfect table tradeoff algorithms analyzed in this work may also be roughly compared against the three non-perfect table tradeoff algorithms.

### 5.2 Comparisons

Since we are going to compare the tradeoff algorithms at a common fixed probability of success, the parameters $\bar{\mathtt{D}}_{\mathrm{ps}}$ and $\bar{\mathtt{R}}_{\mathrm{ps}}$ will now be treated as fixed constants. The discussions of Section 3.2 and Section 4.2 show that perfect versions of the DP and rainbow tradeoffs require approximately the same number of bits per table entry as their non-perfect counterparts running under comparable resources. One can conclude that a fair comparison of online efficiencies would compare the adjusted tradeoff coefficients $\frac{1}{4}\bar{\mathtt{D}}_{\mathrm{tc}}$ and $\bar{\mathtt{R}}_{\mathrm{tc}}$ against each other, and that these two adjusted tradeoff coefficients may also be directly compared against the non-perfect rainbow tradeoff coefficient $\mathtt{R}_{\mathrm{tc}}$.

Let us explain how one may plot the pre-computation coefficient versus tradeoff coefficient curves for the two perfect tradeoff algorithms. One can derive

$$\bar{\mathtt{D}}_{\mathrm{pc}} = \frac{2+\bar{\mathtt{D}}_{\mathrm{msc}}}{2\bar{\mathtt{D}}_{\mathrm{cr}}}\big\{-\ln(1-\bar{\mathtt{D}}_{\mathrm{ps}})\big\} \tag{26}$$

from Proposition 4, and since Proposition 8 expresses $\bar{\mathtt{D}}_{\mathrm{cr}}$ as a function of $\bar{\mathtt{D}}_{\mathrm{msc}}$, the pre-computation coefficient can be seen as a function of the single parameter $\bar{\mathtt{D}}_{\mathrm{msc}}$, as long as $\bar{\mathtt{D}}_{\mathrm{ps}}$ is treated as a constant. Similarly, Theorem 10 and Proposition 8 express the tradeoff coefficient $\bar{\mathtt{D}}_{\mathrm{tc}}$ as a function of the single parameter $\bar{\mathtt{D}}_{\mathrm{msc}}$. Thus the relation between $\bar{\mathtt{D}}_{\mathrm{pc}}$ and $\bar{\mathtt{D}}_{\mathrm{tc}}$ may be drawn as a curve parameterized by $\bar{\mathtt{D}}_{\mathrm{msc}}$.

It is important to understand that, even when the success rate $\bar{\mathtt{D}}_{\mathrm{ps}}$ and curve parameter $\bar{\mathtt{D}}_{\mathrm{msc}}$ are fixed to specific values, there still remains a single degree of freedom in the tradeoff algorithm parameters $m$, $t$, and $\ell$, with which one can realize the tradeoff between the online time $T$ and the storage requirement $M$. That is, the ability of the DP algorithm to provide tradeoffs between online time and storage requirement is unimpaired by restrictions on the success rate and the matrix stopping constant, and the $\bar{\mathtt{D}}_{\mathrm{tc}}$ value appearing with each $(\bar{\mathtt{D}}_{\mathrm{pc}}, \bar{\mathtt{D}}_{\mathrm{tc}})$-pair represents an online efficiency with a fully operational time memory tradeoff opportunity.

To be more concrete, suppose one is given specific $\bar{\mathtt{D}}_{\mathrm{ps}}$ and $\bar{\mathtt{D}}_{\mathrm{msc}}$ values, together with any $T$ and $M$ that satisfy the tradeoff curve of Theorem 10, where the tradeoff coefficient $\bar{\mathtt{D}}_{\mathrm{tc}}$ is computed from the given $\bar{\mathtt{D}}_{\mathrm{ps}}$ and $\bar{\mathtt{D}}_{\mathrm{msc}}$ values. Then it is easy to check that the sequentially defined parameter set

$$t = \left(\frac{\bar{\mathtt{D}}_{\mathrm{msc}}\bar{\mathtt{D}}_{\mathrm{cr}}}{\bar{\mathtt{D}}_{\mathrm{ps}}}\left\{1+2\,\mathrm{Li}_2\left(\frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2+\bar{\mathtt{D}}_{\mathrm{msc}}}\right)\right\}^{-1} T\right)^{\frac{1}{2}}, \tag{27}$$

$$m = \frac{\bar{\mathtt{D}}_{\mathrm{msc}}\mathtt{N}}{t^2}, \tag{28}$$

$$\ell = \frac{\mathtt{N}}{mt\bar{\mathtt{D}}_{\mathrm{cr}}}\big\{-\ln(1-\bar{\mathtt{D}}_{\mathrm{ps}})\big\} \tag{29}$$

satisfies the four requests or restrictions on $\bar{\mathtt{D}}_{\mathrm{ps}}$, $\bar{\mathtt{D}}_{\mathrm{msc}}$, $T$, and $M$. The equivalence of (29) and (14) implies that the success rate $\bar{\mathtt{D}}_{\mathrm{ps}}$ will be achieved with these parameters, while (28) ensures that the given $\bar{\mathtt{D}}_{\mathrm{msc}}$ value is adhered to. Furthermore, since (27) and the first equation in the proof of Theorem 10 are equivalent, the online phase is expected to terminate in the

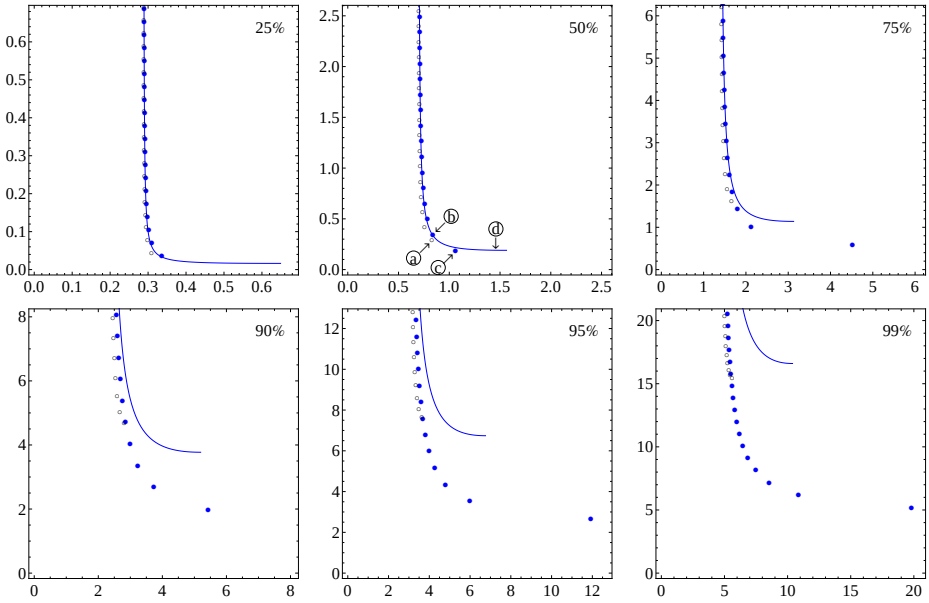


**Fig. 2** The tradeoff coefficients $\frac{1}{4}\bar{\mathtt{D}}_{\mathrm{tc}}$ (line), $\bar{\mathtt{R}}_{\mathrm{tc}}$ (dots), and $\mathtt{R}_{\mathrm{tc}}$ (circles), with adjustments suitable for direct comparison, in relation to their respective pre-computation costs, at various success rates ($x$-axis: $\bar{\mathtt{D}}_{\mathrm{pc}}$, $\bar{\mathtt{R}}_{\mathrm{pc}}$, and $\mathtt{R}_{\mathrm{pc}}$; $y$-axis: $\frac{1}{4}\bar{\mathtt{D}}_{\mathrm{tc}}$, $\bar{\mathtt{R}}_{\mathrm{tc}}$, and $\mathtt{R}_{\mathrm{tc}}$).

requested time $T$. Finally, since both the storage requirement under the above parameter set and the requested $M$ value satisfy the same tradeoff curve, i.e., with common values of the online time and tradeoff coefficient, adherence to $M$ is guaranteed.

Let us now explain how the performance curve for the the perfect rainbow tradeoff may be plotted. One can combine (23) and Proposition 15 to express the pre-computation coefficient

$$\bar{\mathtt{R}}_{\mathrm{pc}} = \frac{2}{2-\bar{\mathtt{R}}_{\mathrm{msc}}}\big\{-\ln(1-\bar{\mathtt{R}}_{\mathrm{ps}})\big\} = -\frac{2\ell\ln(1-\bar{\mathtt{R}}_{\mathrm{ps}})}{2\ell+\ln(1-\bar{\mathtt{R}}_{\mathrm{ps}})} \tag{30}$$

as a function of the single variable $\ell$. Similarly, the substitution of (23) into the formula of Theorem 17 results in an expression for $\bar{\mathtt{R}}_{\mathrm{tc}}$ that is given in terms of the single variable $\ell$. Thus, the relation between $\bar{\mathtt{R}}_{\mathrm{pc}}$ and $\bar{\mathtt{R}}_{\mathrm{tc}}$ may be drawn as a curve parameterized by $\ell$. As with the DP tradeoff, possibility of tradeoffs between online time and storage requirement remains unaffected by the restrictions on $\bar{\mathtt{R}}_{\mathrm{ps}}$ and $\ell$.

The curves for the two perfect tradeoffs and the non-perfect rainbow tradeoff are given in Figure 2 for some specific success rates. Within each box, being lower is to have better online efficiency and being closer to the left edge is to require less pre-computation. Hence, one may roughly interpret being situated closer to the lower left corner as displaying better performance.

In each box, the filled dots represent the choice of $(\bar{\mathtt{R}}_{\mathrm{pc}}, \bar{\mathtt{R}}_{\mathrm{tc}})$-pairs made available by the perfect rainbow tradeoff, and the empty circles represent data for the non-perfect rainbow tradeoff. Each dot for the perfect rainbow tradeoff corresponds to an integer $\ell$ value, with the rightmost dot corresponding to $\ell = \lceil -\frac{1}{2}\ln(1-\bar{\mathtt{R}}_{\mathrm{ps}})\rceil$, as determined by the bound (24). Since the table count $\ell$ must be an integer, the available choices appear as a discrete set of points. Similar statements may be made for the circles that represent data for the non-perfect

rainbow tradeoff. The single continuous curve in each box represents data for the perfect DP tradeoff. Unless $\mathsf{N}$ is small, it is reasonable to treat the parameter $\bar{\mathsf{D}}_{\mathrm{msc}} = \frac{mt^2}{\mathsf{N}}$ that was used to draw these graphs as a continuous variable, even though it originates from integers. One can numerically verify that the tradeoff coefficient $\bar{\mathsf{D}}_{\mathrm{tc}}$ attains its minimum at $\bar{\mathsf{D}}_{\mathrm{msc}} = 1.41349$, regardless of the $\bar{\mathsf{D}}_{\mathrm{ps}}$ value. The lowest point, or the right end, of the curve in each box corresponds to $\bar{\mathsf{D}}_{\mathrm{msc}} = 1.41349$ and the curve is drawn only for the parameter in the range $\bar{\mathsf{D}}_{\mathrm{msc}} \leq 1.41349$. Curve points corresponding to larger $\bar{\mathsf{D}}_{\mathrm{msc}}$ values would be associated with worse online efficiencies at higher pre-computation costs, which would not be used.

Let us first focus our attention on the box labeled 50%. A careful close view shows that, compared to point ⓑ, the point ⓐ is associated with better online efficiency at a very slightly lower pre-computation cost. Hence, any tradeoff implementer would prefer the non-perfect rainbow option ⓐ over the perfect rainbow option ⓑ. On the other hand, the preferences between the non-perfect rainbow option ⓐ and the perfect rainbow option ⓒ will be different for each implementer and at every situation, and neither option can be said to be at a clear advantage over the other. When the overall distribution is considered, since the circles are closer to the bottom left corner than the filled dots, one may claim that the non-perfect rainbow tradeoff is very slightly better than the perfect rainbow tradeoff. On the other hand, one may still plausibly claim that the perfect rainbow tradeoff is advantageous in that it provides one extra option, namely ⓒ, that cannot be approximately provided by the non-perfect rainbow tradeoff. As for the perfect DP tradeoff, even though its overall performance is very close to those of the two rainbow tradeoffs, it could be preferred over the two rainbow tradeoffs, since it allows for more flexibility in options by filling in the spaces left open by the discrete set of dots and circles. Some might be tempted to claim that the curve point at ⓓ is a perfect DP tradeoff option that cannot be provided by the two rainbow tradeoffs, but this option is not useful, since it corresponds to online efficiency similar to ⓒ at a much higher pre-computation cost. In general, it seems fair to claim that, when the requirement on the success rate is low, the performances of the perfect rainbow, non-perfect rainbow, and perfect DP tradeoffs are quite similar, with the perfect DP tradeoff at a slight advantage in view of its flexibility in options.

The situation is clearly different at higher success rate requirements, where the perfect DP tradeoff is no match for the perfect rainbow tradeoff. In the case of 99% success rate, the perfect DP tradeoff is clearly less desirable than even the non-perfect rainbow tradeoff, regardless of how one wants to balance online efficiency against pre-computation cost. It also seems fair to claim that the perfect rainbow tradeoff is at an advantage over the non-perfect rainbow tradeoff, since it can approximately provide every option made available by the non-perfect rainbow, while providing many more options that cannot be approximated by the non-perfect rainbow tradeoff. Furthermore, the perfect rainbow tradeoff presents the possibility of obtaining much better online efficiencies, although these must be paid for with higher pre-computation costs.

The comments given so far may seem generally plausible, but when lowering the pre-computation cost is immensely important, there remains a small possibility that the non-perfect rainbow tradeoff could be preferred over the perfect rainbow tradeoff. This is illustrated by Figure 3, which is an enlarged view of a narrow vertical strip from the 99% box of Figure 2. We have intentionally stretched the vertical strip in the horizontal direction and have reduced the hight, so that even a small difference in the pre-computation coefficient is felt as being significant. Even though some sacrifice in the online efficiency is inevitable, the options provided by the non-perfect rainbow tradeoff now seem much more reasonable than previously felt when viewed from within Figure 2. Note that similar statements may be made

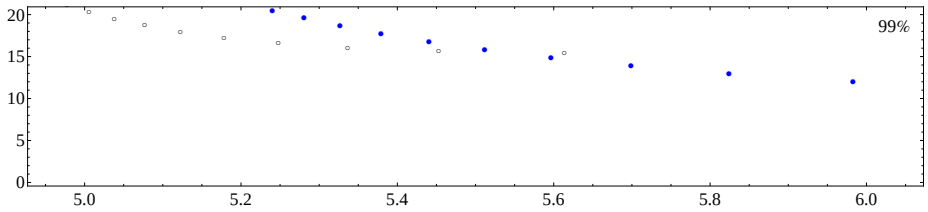


**Fig. 3** Tradeoff coefficient for perfect (dots) and non-perfect (circles) rainbow tradeoffs in relation to their pre-computation costs at 99% success rate ($x$-axis: $\bar{\mathrm{R}}_{\mathrm{pc}}$ and $\mathrm{R}_{\mathrm{pc}}$; $y$-axis: $\bar{\mathrm{R}}_{\mathrm{tc}}$ and $\mathrm{R}_{\mathrm{tc}}$).

for all the boxes. If lowering the pre-computation cost is extremely important, the seemingly small advantage of the non-perfect rainbow tradeoff over the perfect rainbow tradeoff may be of much more value than we had previously given credit to while discussing the 50% box.

To summarize, when the online efficiency and pre-computation cost are both taken into account, the perfect rainbow tradeoff is likely to be advantageous over perfect DP, non-perfect rainbow, non-perfect DP, and classical Hellman tradeoffs, in typical situations. However, there may be special circumstances under which the preferences could be different. For example, importance of lowering the pre-computation cost may shift the preference towards the non-perfect rainbow tradeoff, and the need for fine-tuned parameter choices may make the perfect DP tradeoff, or even the non-perfect DP and classical Hellman tradeoffs, favorable at low success rate requirements.

## 6 Conclusion

In this work, we have analyzed the execution complexity of the perfect DP tradeoff and have computed its tradeoff coefficient. We have also combined the existing results on the execution complexity of the perfect rainbow tradeoff to present its online efficiency. Using this information, the performances of the perfect DP tradeoff and perfect rainbow tradeoff were compared against each other. We also added the non-perfect rainbow tradeoff, which was shown to outperform the classical Hellman and non-perfect DP tradeoffs in a previous work [11], to our comparison. Our conclusion, when overly simplified, was that the perfect rainbow tradeoff is advantageous over the other four tradeoff algorithms.

On the surface, our conclusion may seem to be a repetition of the claim given by the article [12] that introduced the rainbow tradeoff, but the two claims differ fundamentally in their contents. The previous work [12] measured the storage size in terms of the number of table entries, but the physical storage size was considered in our comparisons. Our comparison was based on the expected execution complexities, rather than the worst case complexities, which were used by [12]. Finally, our comparison considered both the online efficiency and the cost of pre-computation, whereas only the optimal online behavior was considered by [12]. In other words, different concepts of one algorithm being better than another algorithm were employed by [12] and this work in comparing tradeoff algorithms.

In addition to the conceptual differences which make our comparisons more meaningful in practice than the previous claims of [12], there was also a difference in the accuracy of the two comparisons. The claim of [12] relied on parameter sets for different algorithms that were heuristically shown to bring about roughly corresponding success rates, but the parameter sets for different algorithms were conditioned to achieve exactly the same success

rate in our comparison. In addition, our comparison, unlike that of [12], fully accounted for the complexity of resolving false alarms, which constitutes a significant portion of the online time.

Despite our simplified conclusion of the perfect rainbow tradeoff outperforming the other four algorithms, we do not rule out the possibility of encountering situations where the conclusions could be different. One example would be when lowering the pre-computation cost is immensely important. There may also be issues we have ignored, such as the possibility of loading a pre-computation table into fast memory or table lookup characteristics, which could make the two DP tradeoffs, with smaller individual tables and relatively uniform interval between table lookups, preferable over the two rainbow tradeoffs. Furthermore, recent implementation platforms such as GPUs, where the scheduling of the online computation needs to be totally different from most computation platforms and accesses to large storage are inconvenient, could call for a new analysis of the execution complexities.

It remains to verify whether any of the recent tradeoff algorithm variants that claim superiority over existing algorithms are truly advantageous over the perfect rainbow tradeoff.

## A Chain Length Bounds and Parallelization of the DP Tradeoff

Let us briefly comment on the two techniques for the DP tradeoff that were not considered during our complexity analyses. Specifically, these were the upper and lower bounds on the chain lengths and the parallel processing of tables during the online phase.

The use of chain length bounds will increase the number of discarded pre-computation chains, so that the amount of pre-computation must be increased in order to maintain the success rate. The effect of chain length bounds on the online efficiency is unclear, but making the pre-computation chain lengths shorter has the tendency to reduce chain merges and false alarms, so this could have a positive effect. However, a quick review of the high success rate boxes in Figure 2 shows that no small enhancement in the online efficiency is likely to make the perfect DP tradeoff more preferable over the perfect rainbow tradeoff, even if no penalty on the pre-computation cost was involved. Hence, there seems to be little reason to consider the perfect DP tradeoff variant that utilizes chain length bounds, except possibly at low success rates.

The work [8, 15] suggests that all the pre-computation tables be processed in parallel, rather than sequentially, during the online phase. Parallel processing causes the shorter online chains to be treated before the longer ones, and since the online phase is likely to terminate with the correct answer before any of the pre-computation tables are fully processed, this leads to a larger portion of the online computation being spent on processing the shorter chains. Since shorter chains are less likely to induce false alarms, this has a positive effect of reducing the cost of dealing with alarms. However, the recent analysis on the non-perfect parallel DP tradeoff [10] indicates that one cannot hope to see any drastic improvement of the perfect DP tradeoff performance through parallelization. For now, there seems to be no reason to expect the parallel perfect DP tradeoff to perform better than the perfect rainbow tradeoff, especially under moderate or high success rate requirements.

## B Detailed Proof of Lemma 5

The very short proof of Lemma 5 given in the main body of this paper will seem sufficient to most cryptographers. However, there are subtle issues involving random functions that were ignored in the proof. This section is an attempt at resolving these issues. We strongly urge any interested reader to review [11, Appendix B] before reading this section.

Recall that we are using $\mathscr{D}_k(F)$ to denote the set of elements of $\mathscr{N}$ that are $k$-many $F$-iterations away from their closest DPs, and suppose that $D_0, D_1, \ldots, D_k$ are any collection of mutually disjoint subsets of $\mathscr{N}$, with $D_0$ denoting the set of all DPs. Let us consider any function $F : \mathscr{N} \to \mathscr{N}$ and discuss when the series of conditions

C1: $D_i = \mathscr{D}_i(F)$, for $i = 1, \ldots, k$

would be satisfied by the function. Note that the omitted condition $D_0 = \mathscr{D}_0(F)$ is always satisfied.

The series of conditions C1 is satisfied by the function $F$ if and only if the following three items are all satisfied:

C2: $D_i = \mathscr{D}_i(F)$, for $i = 1, \ldots, k-1$.
C3: $F(D_k) \subset D_{k-1}$.
C4: $F(\mathscr{N} \setminus D_k) \cap D_{k-1} = \emptyset$.

Repeating this argument, we see that the satisfaction of Condition-C1 by an $F$ is equivalent to the satisfaction of the following two series of conditions.

C5: $F(D_k) \subset D_{k-1}, \ldots, F(D_2) \subset D_1$, and $F(D_1) \subset D_0$.
C6: $F(\mathscr{N} \setminus D_k) \cap D_{k-1} = \emptyset, \ldots, F(\mathscr{N} \setminus D_2) \cap D_1 = \emptyset$, and $F(\mathscr{N} \setminus D_1) \cap D_0 = \emptyset$.

Suppose that we were trying to construct a function that satisfies these conditions. Then the condition $F(D_k) \subset D_{k-1}$ would certainly places a restriction on how we may define $F$ on $D_k$. However, since the set $D_k$ does not overlap with any of the other sets $D_0, \ldots, D_{k-1}$, none of the other sub-conditions of C5 places any restriction on the definition of $F$ on $D_k$. Next, the sub-condition $F(\mathscr{N} \setminus D_k) \cap D_{k-1} = \emptyset$ of C6 is certainly irrelevant to the definition of $F$ on $D_k$. The other sub-conditions of C6 are not quite so irrelevant, but since $D_{k-1}$ does not overlap with any of the sets $D_0, \ldots, D_{k-2}$, as long as the condition $F(D_k) \subset D_{k-1}$ is adhered to, they are automatically satisfied and do not places any additional restriction of how we may define $F$ on $D_k$. More generally, for each $i = 1, \ldots, k$, the condition $F(D_i) \subset D_{i-1}$ is the only restriction placed on the definition of $F$ on $D_i$ by C5 and C6. In other words, asking for C1 to be satisfied by a function $F$ places no restriction on the definition of $F$ on $D_i$ other than that $F(D_i) \subset D_{i-1}$ be satisfied.

The discussion given so far can be reinterpreted as follows. Let $D_0, D_1, \ldots, D_k$ be any collection of mutually disjoint subsets of $\mathscr{N}$, with $D_0$ denoting the set of all DPs. By choosing a function $F : \mathscr{N} \to \mathscr{N}$ satisfying Condition-C1, one determines functions $F_i : D_i \to D_{i-1}$ for each $i = 1, \ldots, k$. Let us fix any $i$. If the choice of $F$ is made uniformly at random from the set of all function satisfying Condition-C1, the distribution of the corresponding $F_i : D_i \to D_{i-1}$, defined on the set of all functions having domain $D_i$ and codomain $D_{i-1}$, also becomes the uniform distribution.

We are now ready to prove the lemma. Choose any explicit family of disjoint sets $D_0, \ldots, D_k$ such that $D_0$ is the set of all DPs. For a random function satisfying Condition-C1, since each $F_i : D_i \to D_{i-1}$ is a random function, we can expect to see

$$\frac{|F(D')|}{|D_{i-1}|} = 1 - \exp\Big(-\frac{|D'|}{|D_{i-1}|}\Big) \tag{31}$$

for every set $D' \subset D_i$ and $i = 1, \ldots, k$. So far, we have fixed $D_0, \ldots, D_k$ first and have claimed information on iterated image sizes only for functions $F$ satisfying Condition-C1, which is associated with the sets $D_0, \ldots, D_k$. However, observe that this iterative equation depends only on the set sizes $|D'|, |D_0|, \ldots, |D_k|$ and not on the specific sets $D', D_0, \ldots, D_k$. Also recall that (15) is expected of a random function. Interpreting this as (15) holding accurately for the vast majority of functions $F$, the truth of Lemma 5 follows. A review of [11, Appendix B] is required to fully understand this final statement.

## C Comments on a Previous Analysis of the perfect DP Tradeoff

Since there is a large overlap between what the previous work [16] and the current work claim to have obtained, let us compare some of the results from the two works.

The paper [16] provides a method for computing the number of distinct ending DPs expected from a given number of starting points. This value, which is the number of chains remaining after removal of merging chains, was referred to as $\alpha$ in Section 2.2. In Table 2, we have copied some of the related data appearing in [16, Table 2] and have appended our corresponding theoretic values, computed with Lemma 2. The parameters associated with this table are $\mathsf{N} = 2^{56}$ and $t = 2^{18}$, and all figures in the table are given as $\log_2$ values. Recall that Lemma 2 is valid for the case when no chain length bounds are set. The data from [16] are for when there is no lower bound on the chain length and the chain length upper bound is set to $2^{30}$, which is sufficiently large for the $t = 2^{18}$ being used.

One can verify, in every row of the table, that our theory provides estimates that are closer to the test results of [16] than their own theory. Some might dismiss the differences between the two theoretic values as being too small to be of any significance, but one must consider the fact that these are $\log_2$ values. For example, referencing the last row of the table, we can see that the (in)accuracy of Lemma 2 in predicting the test value is by a factor of $\frac{2^{21.6425}}{2^{21.6430}} \approx 0.999653$, but that the theoretic prediction of [16] is further away at a

**Table 2** Number of DP chains before and after removal of chain merges ($\mathsf{N} = 2^{56}$, $t = 2^{18}$). All values are given in $\log_2$ scale.

| # of starting points | # of ending points | | |
|---|---|---|---|
| | test of [16] | theory of [16] | Lemma 2 |
| 20 | 19.5497 | 19.4712 | 19.5500 |
| 21 | 20.3048 | 20.1357 | 20.3058 |
| 22 | 20.9990 | 20.6866 | 21.0000 |
| 23 | 21.6425 | 21.1357 | 21.6430 |

factor of $\frac{2^{21.6425}}{2^{21.1357}} \approx 1.42$. The error factor of 1.42 in the estimated number of ending points is directly reflected in the pre-computation table size estimate and presents itself as a $1.42^2 \approx 2.02$ error factor in the estimate for the tradeoff coefficient. Hence, for the purpose of comparing the online efficiencies of tradeoff algorithms, which usually differ only by factors of such small magnitude, the theory of [16] was not accurate enough.

Recall from Section 2.2 that the expression for $\alpha$ given in [16] involved the average chain length $\bar{\beta}$ and that their derivation of $\bar{\beta}$ involved an ad hoc procedure. So, let us consider the possibility that this ad hoc procedure was the cause of the inaccuracy we have witnessed in Table 2 and that their arguments were correct up to that point. For this purpose, we focus on one major intermediate result of [16] which they refer to as the storage function $s(j)$.

The exact definition of $s(j)$ is slightly complicated and will not be explained here in full, but when no bounds are placed on the chain lengths, function $s(j)$ is suppose to give the expected number of distinct nodes in a perfect DP matrix that was created with $j$ starting points. The formula they give is

$$s(j) = 2^k \Big\{ 1 - \Big( \frac{2^k}{-\beta j + \beta^2 j + K} \Big)^{\frac{1}{\beta - 1}} \Big\} \quad \text{with} \quad K = 2^k \Big( 1 - \frac{s(0)}{2^k} \Big)^{1-\beta}, \tag{32}$$

where the $\beta$ they use is the average chain length before the removal of chain merges and $2^k$ is the size of the search space. The presence of $s(0)$ may seem strange, but this is because we have not explained the meaning of $s(j)$ in full, and we may simply take $s(0) = 0$ for this discussion. In the absence of chain length bounds, we may take $\beta = t$ and approximate their formula into

$$s(j) = \mathsf{N} \Big\{ 1 - \Big( \frac{\mathsf{N}}{t^2 j + \mathsf{N}} \Big)^{\frac{1}{t}} \Big\}, \tag{33}$$

written in the notation of this paper. Hence, according to the theory of [16], if one creates a perfect DP matrix using $m_0$ starting points, there will be

$$|\bar{\mathtt{DM}}| = s(m_0) = \mathsf{N} \Big\{ 1 - \Big( \frac{1}{1 + \mathtt{D}_{\mathrm{msc}}} \Big)^{\frac{1}{t}} \Big\} \tag{34}$$

distinct points in the matrix, where $\mathtt{D}_{\mathrm{msc}} = \frac{m_0 t^2}{\mathsf{N}}$. To allow for direct comparison with our result, we rewrite this in the form

$$\frac{t}{\mathsf{N}} |\bar{\mathtt{DM}}| = t \Big\{ 1 - \Big( \frac{1}{1 + \mathtt{D}_{\mathrm{msc}}} \Big)^{\frac{1}{t}} \Big\} \approx \ln(1 + \mathtt{D}_{\mathrm{msc}}). \tag{35}$$

The approximation here is accurate for all sufficiently large $t$. For example, at $\mathtt{D}_{\mathrm{msc}} = 1$ the right-hand value is $\ln(1 + \mathtt{D}_{\mathrm{msc}}) = 1.09861$, and even at the moderately sized $t = 2^{10}$, the formula in the middle involving $t$ evaluates to 1.09802.

Our corresponding result is contained in Proposition 8. Using (12), we can claim

$$\frac{t}{\mathsf{N}} |\bar{\mathtt{DM}}| = \bar{\mathtt{D}}_{\mathrm{msc}} \bar{\mathtt{D}}_{\mathrm{cr}} = 2 \ln \Big( 1 + \frac{\bar{\mathtt{D}}_{\mathrm{msc}}}{2} \Big) = 2 \ln \Big( \frac{1 + \sqrt{1 + 2\mathtt{D}_{\mathrm{msc}}}}{2} \Big), \tag{36}$$

which is clearly different from (35).

The numeric values given by the two formulas (35) and (36) are compared in Figure 4. The two are visibly different, except when $\mathtt{D}_{\mathrm{msc}}$ is very small, in which case chain merges are rare and both theories reduce to something trivial. For example, even at $\mathtt{D}_{\mathrm{msc}} = 1$, the two theories give clearly different values of 0.69315 and 0.62381. Since the testing of Section 3.3 has shown our estimate of coverage rate to be much more accurate than the general order of these differences, the storage function $s(j)$, as given by (33), must not be as accurate.

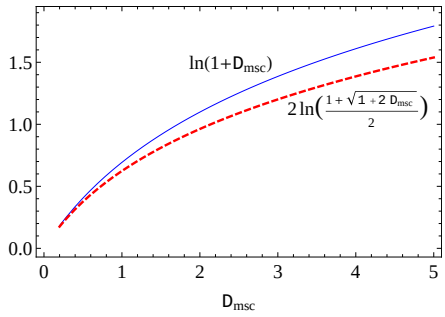


**Fig. 4** The value $\frac{t}{\mathsf{N}}|\tilde{\mathbb{D}}\tilde{\mathbb{M}}| = \tilde{\mathsf{D}}_{\mathrm{msc}}\tilde{\mathsf{D}}_{\mathrm{cr}}$ as predicted by [16] (*solid*) and the current work (*dashed*).

Let us explain the source of their inaccuracy. The core logic of [16] rests in their Equation (22), which we approximate and state as

$$s(j+1) = s(j) + t\left(1 - \frac{s(j)}{\mathsf{N}}\right)^t, \tag{37}$$

for the case when there are no chain length bounds. This equation can be interpreted as follows. One expects to add $t$ new points to the perfect matrix by considering one more pre-computation chain, but this addition should only be done when the chain of $t$ points do not merge with the points within the existing matrix. The $t$-th powered term on the right is the probability for none of the $t$ additionally generated points to be in the existing matrix of $s(j)$ points. This high-level view of (37) may seem plausible.

The first source of inaccuracy hiding in (37) is the assumption that the new chain added is always of length $t$. In reality, chains of varying lengths will be created and these all have different probabilities of merging with the existing matrix. The simplified view of [16] could have been justified to a certain degree if the chain lengths were mostly close to $t$, but the actual distribution of chain lengths is not even centered at the average value $t$. The authors of [16] seem to have been aware of this problem. In later parts of their paper, they divide the range of possible chain lengths into a small number of segments and treat each length range separately, using different average chain lengths within each segment.

The second source of inaccuracy is in the $t$-th powered term that was suppose to give the probability for the additional chain not to merge with the existing matrix. Since we are dealing with a DP chain of a predetermined length at this point, a more accurate expression would somehow involve the set sizes $|\mathscr{D}_k|$, that appeared in the main body of this work, in the denominator, rather than $\mathsf{N}$. Furthermore, the numerator $s(j)$ should be replaced with values that are associated in some way with the *non-perfect* matrix created up to that point. This is because the newly created chain is destined to merge with the perfect matrix whenever it merges with a previously generated chain, regardless of whether the chain has been discarded due to merging.

We conclude that while the analyses of [16] are base on plausible arguments, their results are valuable only as first approximations and are not accurate enough for the purpose of this paper, which is to compare the performances of different tradeoff algorithms.

## D Optimal Rainbow Tradeoff Parameters

In this section we discuss the parameter set for the perfect rainbow tradeoff that the previous analysis [1] suggested and claimed to be optimal. Let us explain the steps they take to fix the parameters, in the language of this work. The required success rate $\bar{\mathsf{R}}_{\mathrm{ps}}$ and the total number of table entries $M$ are treated as the externally supplied parameters. First (24) is used to fix the number of tables $\ell = \lceil -\frac{1}{2}\ln(1-\bar{\mathsf{R}}_{\mathrm{ps}})\rceil$. This forces the number of starting points per table to $m = \frac{M}{\ell}$ and we know from (22) that the remaining parameter $t$ must be fixed to

$$t = -\frac{\mathsf{N}}{m\ell}\ln(1-\bar{\mathsf{R}}_{\mathrm{ps}}) = -\frac{\mathsf{N}}{M}\ln(1-\bar{\mathsf{R}}_{\mathrm{ps}}), \tag{38}$$

if the requested success rate is to be achieved. In short, the number of tables is chosen to be as small as possible, subject to the condition (24), and the rest of the parameters are set to what they must be in order to satisfy the externally given requirements.

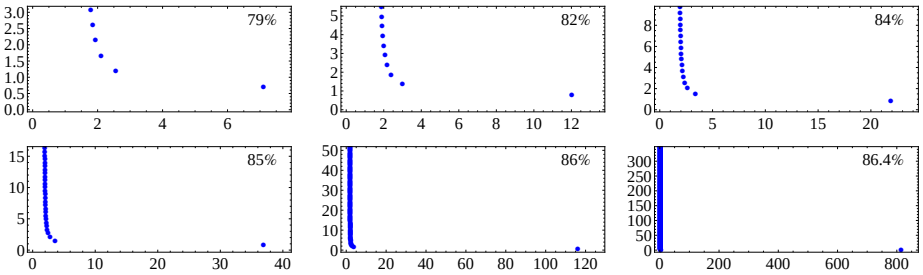


**Fig. 5** Pre-computation cost versus tradeoff coefficient for the perfect rainbow tradeoff at success rates slightly lower than $1 - \frac{1}{e^2} = 86.466\%$ ($x$-axis: $\bar{\mathrm{R}}_{\mathrm{pc}}$; $y$-axis: $\bar{\mathrm{R}}_{\mathrm{tc}}$).

It is not hard to show that the tradeoff coefficient $\bar{\mathrm{R}}_{\mathrm{tc}}$ of Theorem 17, when combined with (23), is an increasing function of $\ell$ in the range $\ell \geq 1$ and $0 \leq \bar{\mathrm{R}}_{\mathrm{ps}} < 1$. This implies that the parameters suggested by [1] are optimal in the sense that it gives the smallest possible tradeoff coefficient among those achieving the intended success rate. An easier approach would be to state that, the optimal parameter set of [1] corresponds to the rightmost filled dot in each box of Figure 2, which is clearly the lowest dot in each box.

However, we want to emphasize that the rightmost dot of each box is just one of the many options that are available to the tradeoff algorithm implementer, and that this specific option is not likely to be favored over others when the pre-computation cost is taken into account, except possibly at low success rates. Furthermore, Figure 5 demonstrates that there are success rates for which using parameters corresponding to the rightmost dot is certainly impractical.

The parameter set that is most efficient in terms of online resource requirements is very often not the most practical or reasonable parameter set option among those made available by a tradeoff algorithm. The current work provides the information which enables the tradeoff algorithm implementer to decide on the most sensible, rather than the most online-efficient, parameter to use.

## References

1. G. Avoine, P. Junod, P. Oechslin, Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Trans. Inform. Syst. Secur.* **11**(4), 17:1–17:22 (2008). Preliminary version presented at INDOCRYPT 2005.
2. E. Barkan, E. Biham, A. Shamir, Rigorous bounds on cryptanalytic time/memory tradeoffs, in *Advances in Cryptology—CRYPTO 2006*. LNCS **4117**, (Springer, 2006), pp. 1–21.
3. A. Biryukov, A. Shamir, D. Wagner, Real time cryptanalysis of A5/1 on a PC, in *FSE 2000*. LNCS **1978**, (Springer, 2001), pp. 1–18.
4. J. Borst, *Block Ciphers: Design, Analysis, and Side-Channel Analysis*. Ph.D. Thesis, Katholieke Universiteit Leuven, September 2001.
5. J. Borst, B. Preneel, J. Vandewalle, On the time-memory tradeoff between exhaustive key search and table precomputation, in *Proceedings of the 19th Symposium on Information Theory in the Benelux*. (WIC, 1998), pp. 111–118.
6. D. E. Denning, *Cryptography and Data Security* (Addison-Wesley, Reading, 1982).
7. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory* **26**, pp. 401–406 (1980).
8. Y. Z. Hoch, Security analysis of generic iterated hash functions. Ph.D. Thesis, Weizmann Institute of Science, August 2009.
9. J. Hong, The cost of false alarms in Hellman and rainbow tradeoffs. *Des. Codes Cryptogr.* **57**, pp. 293–327 (2010).
10. J. Hong, G. W. Lee, D. Ma, Analysis of the parallel distinguished point tradeoff, in *Progress in Cryptology—INDOCRYPT 2011*. LNCS **7107**, (Springer, 2011), pp. 161–180.
11. J. Hong, S. Moon, A comparison of cryptanalytic tradeoff algorithms. To appear in *J. Cryptology*. Published online on July 2012 (`http://dx.doi.org/10.1007/s00145-012-9128-3`). Earlier version available from the Cryptology ePrint Archive as Report 2010/176.
12. P. Oechslin, Making a faster cryptanalytic time-memory trade-off, in *Advances in Cryptology—CRYPTO 2003*. LNCS **2729**, (Springer, 2003), pp. 617–630.

13. J.-J. Quisquater, J. Stern, Time-Memory Tradeoff Revisited. Unpublished, December 1998.
14. N. Saran, *Time Memory Trade Off Attack on Symmetric Ciphers*. Ph.D. Thesis, Middle East Technical University, February 2009.
15. A. Shamir, Random Graphs in Security and Privacy. Invited talk at ICITS 2009.
16. F.-X. Standaert, G. Rouvroy, J.-J. Quisquater, J.-D. Legat, A time-memory tradeoff using distinguished points: New analysis & FPGA results, in *Cryptographic Hardware and Embedded Systems—CHES 2002*. LNCS **2523**, (Springer, 2003), pp. 593–609.
17. M. J. Wiener, The full cost of cryptanalytic attacks. *J. Cryptology* **17**, pp. 105–124 (2004).