manuscript No.
(will be inserted by the editor)

A Comparison of Cryptanalytic Tradeoff Algorithms

Jin Hong · **Sunghwan Moon**

Abstract The three major time memory tradeoff algorithms are compared in this paper. Specifically, the Hellman tradeoff algorithm, the distinguished point tradeoff method, and the rainbow table method, in their non-perfect table versions, are considered.

We show that, under parameters that are typically considered in theoretic discussions of the tradeoff algorithms, Hellman and distinguished point tradeoffs perform very close to each other and the rainbow table method performs somewhat better than the other two algorithms. Our method of comparison can easily be applied to other situations, where the conclusions could be different.

The analysis presented in this paper takes the effects of false alarms into account and also fully considers techniques for reducing storage, such as the ending point truncation method and index files.

Keywords time memory tradeoff · Hellman · distinguished point · rainbow table · random function

1 Introduction

There are numerous security systems in use today that depend on passwords. Access to many contents on the network requires one to login with a password and many file formats today have security features that restrict access to the file until the correct password is supplied. These systems are usually based on a *password hash* technique, which is to store a one-way

Contact author: Jin Hong

JH was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0004561).

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

S. Moon
Department of Mathematics, Texas A&M University, College Station, TX 77843-3368, USA
E-mail: shmoon@math.tamu.edu

function image of the password in the system or file. Indeed, storing the password in its raw form within the file one wishes to set access control to would be meaningless.

A time memory tradeoff algorithm attempts to recover the password from the knowledge of the one-way function image, with the help of a table created through pre-computation. A realistic barrier to applying the tradeoff technique to any specific security system is the massive pre-computation required, before the actual attack can be mounted. The pre-computation cost is roughly proportional to the space of possible passwords and, since most users do not use long or random passwords, the tradeoff attacker is free to choose a smaller subset consisting of short or more likely passwords and decide to be satisfied with recovering only those passwords belonging to this subset. Then the pre-computation requirement no longer stands as a barrier to the tradeoff attack.

It has long been known that properly *salting* a password can remove any realistic threats from time memory tradeoff attacks. However, such measures are still not being taken by many proprietary systems and some systems are known to be using both the newer salted and the older non-salted versions of the security system simultaneously to remain compatible with older systems. Hence, the time memory tradeoff technique still remains a powerful tool against these vulnerable password hash systems. Since human generated passwords will continue to be used for the foreseeable future, one would like to fully understand the power and limitations of the tradeoff techniques.

In this work, we consider the three major tradeoff algorithms. These are the original tradeoff algorithm [8] devised by Hellman, the distinguished points method, attributed to Rivest in [4, p.100], and the rainbow table method [13], announced by Oechslin. There are what is known as the *perfect table* versions of these methods and, even though they are more efficient in recovering passwords, they require more pre-computation. These are hence less practical and we shall not consider them in this paper. There are also multi-target versions of tradeoff algorithms [3, 7] which attracted attention as attacks on streamciphers, but we will not work with these either. The most practical application of the tradeoff technique today is with password hash systems and we will present the current work with this application in mind.

It has been shown [2] that the tradeoff algorithms that are known today already achieve the best asymptotic performance one can hope to reach, if we restrict ourselves to a certain reasonable class of algorithms. However, the measure of performance considered by this theory is only accurate up to a small multiplicative factor. The relative performance of different tradeoff algorithms has not yet been accurately analyzed, and the choice of which algorithm to use can be a difficult one for someone new to the tradeoff technique. In practice, experience seems to be a critical factor in making decisions, and researchers seem to have varied opinions on which algorithm performs better.

Comparison of tradeoff algorithms has been a controversial subject. There are two major obstacles to providing a fair comparison of tradeoff algorithms. The first is that the online time of each algorithm is hard to predict accurately, due to the effects of events called false alarms. Some answers to this problem may now be found in [1, 9] for the Hellman and rainbow cases. The second obstacle concerns the number of bits that is required to store each table entry resulting from the pre-computation. There is a technique for reducing this number of bits called ending point truncation which has not been fully analyzed. Due to these obstacles, previous comparisons of tradeoff algorithms have mostly relied on rough arguments that emphasize a certain advantageous characteristic of one algorithm against another.

There is a naturally occurring measure[1] of how efficiently a tradeoff algorithm balances time against memory and this measure becomes accessible once the first obstacle mentioned above is resolved. In this work, we carefully note that this measure of tradeoff efficiency is expressed in different units for different algorithms and provide arguments for converting them to directly comparable units. This transition of units is intimately connected to the second obstacle mentioned above.

Apart from the above two obstacles that are due to our lack of knowledge, there is yet another problem of how to compare different tradeoff performances that are achievable only after different pre-computation efforts.

In this work, we clear all of the obstacles mentioned so far and provide fair comparisons between tradeoff algorithms. More precisely, we shall present a method to visualize what can be achieved by each algorithm in terms of pre-computation and tradeoff performance. This will be done in a unified way so that the range of choices possible with each algorithm can directly be compared against each other. Any potential user of the tradeoff technique can use this information to decide on which algorithm to use and which set of parameters to use with the algorithm. The judgement of which algorithm is more suitable depends on how the user values the pre-computation cost and tradeoff efficiency relative to the other, and hence cannot be done in an objective manner in most cases.

In presenting the above comparison method, we will mainly focus on a certain set of parameters that is typically considered during theoretic analysis of tradeoff algorithms. Under this parameter set, the Hellman and distinguish point methods are shown to perform very close to each other, while the rainbow table method shows better performance than the other two algorithms, under the additional assumption that the success rate requirement placed on the tradeoff algorithms is rather high. In this specific case, the comparison judgement will stand true for any reasonable way of valuing the pre-computation cost and tradeoff efficiency. Conclusions at other parameter ranges, which may be suitable for many situations, can be different.

Another contribution of this paper is in making the basis of tradeoff technique analysis theoretically more concrete. We discuss the use of the term *random function* and how it is related to the analysis of the tradeoff technique. We identify a common argument that is not mathematically correct in the strict sense and provide plausible justifications for still using such an argument in the analysis of tradeoff algorithms.

The rest of the paper is organized as follows. We briefly fix notation and terminologies in the next section. The use of random functions in the analysis of the tradeoff algorithms is discussed in Section 3. This is followed by a section clarifying the connection between the theory of tradeoff algorithms and its application to password hash systems. In Section 5, 6, and 7, we study the distinguished point, Hellman, and rainbow table tradeoff algorithms, in turn. For each algorithm, we present an accurate version of the tradeoff curve that does not ignore small multiplicative factors and also analyze applicable storage reduction techniques. Comparison of tradeoff performances under different parameter sets for the same algorithm is discussed in Section 8, and performance comparison between different algorithms is given in Section 9. Finally, the work is summarized and concluding remarks are given in Section 10. Experiment data supporting much of the arguments of this paper are given in Appendix C. We acknowledge that a small part of this work was previously made public through [12].

[1] The optimal value of this measure is referred to as the tradeoff characteristic in [1], where it is used to compare the perfect version of the rainbow table method against other algorithms.

2 Time Memory Tradeoff Algorithms

We ask readers to refer to the original papers on the Hellman tradeoff [8] and the rainbow table method [13] for the details of the tradeoff algorithms. Here, we will only list and fix the basic terminologies. Notation of this section will be used throughout the paper. From this point on, we shall refer to the tradeoff algorithm that integrates the distinguished point idea into the Hellman tradeoffs as the DP tradeoff and refer to the rainbow table method simply as the rainbow tradeoff.

A *Hellman chain*, associated with a one-way function $F : \mathcal{N} \to \mathcal{N}$, is of the form

$$\mathbf{sp}_i = \mathbf{x}_{i,0} \xrightarrow{F} \mathbf{x}_{i,1} \xrightarrow{F} \cdots \xrightarrow{F} \mathbf{x}_{i,t} = \mathbf{ep}_i \quad (1 \le i \le m).$$

The *Hellman table* $\{(\mathbf{sp}_i, \mathbf{ep}_i)\}_{i=1}^m$, consisting of starting point and ending point pairs, is sorted on the ending points to make table lookups easier. We have omitted the *reduction functions*, as this will only be mentioned briefly in Section 4.3. The complete set of m chains, consisting of m *rows* and $t+1$ *columns*, is a *Hellman matrix*. The original Hellman tradeoff specified the matrix stopping rule $mt^2 = |\mathcal{N}|$ and considered the use of exactly t tables, but we shall allow more flexibility. We let the positive integer parameters m and t satisfy $mt^2 = \mathtt{H}_{msr}|\mathcal{N}|$, with the positive constant $\mathtt{H}_{msr}$ neither very large nor too close to zero. The condition on $\mathtt{H}_{msr}$ may also be expressed as $\mathtt{H}_{msr} = \Theta(1)$. The number of tables is set to $l = \mathtt{H}_{nt}t$, where $\mathtt{H}_{nt} = \Theta(1)$, so that l and t are roughly of the same order.

A *DP chain* is a chain iteratively constructed by applying $F : \mathcal{N} \to \mathcal{N}$ until a *distinguished point* (DP) is reached. Although not absolutely necessary, to simplify our later discussions, we shall assume that the starting points are always chosen among non-DPs. Hence, in a DP chain, every point before the ending point, including the starting point, is a non-DP. A rigorous treatment that allows starting points to be DPs can be done, but differences between results from such an analysis and those presented in this work will be negligible.

We assume that the distinguishing property is defined in such a way that a random point of $\mathcal{N}$ satisfies it with probability $\frac{1}{t}$. To detect chains that fall into infinite loops, a chain length bound of $\hat{t}$ is fixed and any chain that fails to reach a DP by this length during either the pre-computation phase or the online phase is discarded. Chains are generated until each table contains approximately m chains. What we mean by approximately m will be made more explicit at the beginning of Section 5.

Even though some of our result statements will display its dependence on $\hat{t}$, we shall mainly be interested in the limiting case where $\hat{t}$ is sufficiently large. When $\hat{t} \gg t$, the number of discarded chains is minimized, so that any additional pre-computation is more efficiently transformed into higher success rate of the tradeoff algorithm. Since pre-computation cost is the main barrier to any large scale application of the tradeoff technique, such a choice is only natural in practical applications.

The terms *DP table* refers to the pre-computed table consisting of the starting point and ending point pairs. The term *DP matrix* will be used, even though the collection of DP chains of variable lengths can no longer be visualized as a rectangular matrix. The positive integer parameters m and t are chosen to satisfy $mt^2 = \mathtt{D}_{msr}|\mathcal{N}|$, with the positive constant satisfying $\mathtt{D}_{msr} = \Theta(1)$. The DP tradeoff will use $l = \mathtt{D}_{nt}t$ tables with $\mathtt{D}_{nt} = \Theta(1)$.

We similarly have the notions of *rainbow chain*

$$\mathbf{sp}_i = \mathbf{x}_{i,0} \xrightarrow{F_1} \mathbf{x}_{i,1} \xrightarrow{F_2} \cdots \xrightarrow{F_t} \mathbf{x}_{i,t} = \mathbf{ep}_i \quad (1 \le i \le m),$$

rainbow table, and *rainbow matrix*. For rainbow tradeoffs, it is usual to take $mt = \mathtt{R}_{msr}|\mathscr{N}|$ with $\mathtt{R}_{msr} = \Theta(1)$. A rainbow tradeoff will use l tables, where l is a small positive integer.

Our *inversion target* will always be written as $\mathbf{y} = F(\mathbf{x})$ throughout the paper. The input $\mathbf{x}$ and output $\mathbf{y}$ will be referred to as the *password* and *password hash*, respectively, even though they may not be related to any password based security system.

If the current end $F^k(\mathbf{y})$ of the Hellman tradeoff's *online chain*

$$\mathbf{y} \xrightarrow{F} F(\mathbf{y}) \xrightarrow{F} F^2(\mathbf{y}) \xrightarrow{F} \cdots \xrightarrow{F} F^k(\mathbf{y})$$

of length k matches an ending point $\mathbf{ep}_i$, we have an *alarm*. If an alarm involving $\mathbf{ep}_i$ shows the property $F^{t-k-1}(\mathbf{sp}_i) \neq \mathbf{x}$, it is said to be a *false alarm*. Notice that $F^{t-k-1}(\mathbf{sp}_i) = \mathbf{x}$ is not guaranteed by the weaker condition $F^{t-k}(\mathbf{sp}_i) = \mathbf{y}$. A false alarm is caused by a *merge* between the online chain and a *pre-computed chain*. There notion of false alarms is also used with DP and rainbow tradeoffs.

Throughout this paper, the one-way function $F : \mathscr{N} \to \mathscr{N}$ being considered will always act on a set $\mathscr{N}$ of size N. The parameters m and t satisfying the appropriate $mt^2 = \mathtt{H}_{msr}\mathsf{N}$, $mt^2 = \mathtt{D}_{msr}\mathsf{N}$, or $mt = \mathtt{R}_{msr}\mathsf{N}$ are assumed to be reasonable in the sense that $1 \ll m, t \ll \mathsf{N}$.

There are perfect table versions of the DP and rainbow tradeoffs. These use tradeoff tables that are free of overlapping ending points by removing and regenerating any colliding chains during pre-computation. Such variants are interesting and can be more efficient than the non-perfect versions during the online phase, but require more pre-computation to arrive at the same success rate. As we want to focus on the practical applications of the tradeoff algorithms, we shall not consider perfect tables in this work.

3 Random Functions

There are many cryptographic arguments in the literature that involve *random functions*. In this section, we discuss the extent to which one of these arguments is logical. Although the focus of this paper is in practical applications of the tradeoff technique, this section will only be of theoretic interest. Cryptographers that are comfortable with the term random function will have no trouble in understanding the rest of this paper without reading this section and this section will mainly be of interest to mathematically oriented cryptographers.

Throughout this section, $\mathscr{A}$ and $\mathscr{B}$ will be finite sets of respective sizes A and B. The set of all functions $F : \mathscr{A} \to \mathscr{B}$ will be denoted by $\mathscr{B}^{\mathscr{A}}$. The exponent notation is meant to call to mind the set theoretic construction of a function $F : \mathscr{A} \to \mathscr{B}$ as an ordered A-tuple of elements from $\mathscr{B}$, i.e., an element of the A-times cartesian product of $\mathscr{B}$.

3.1 Definition of a random function

The notion of random functions seems to be one of the most basic concepts used by cryptographers, but a textbook or formally published paper that explains the term in a way that is satisfactory to a mathematician is hard[2] to find. In the interest of making this seemly textbook-level material more widely accessible, we present it here in detail.

[2] It took the authors of this paper quite a long time to *reverse-engineer* the correct meaning of the term random function from its many usages in the literature. It was only after this was done that we were able to ask the right questions to colleagues and were lead to a reference that confirmed our understandings. The lecture note [6, Section 5.2] briefly states the core of what is presented in this subsection.

Before discussing random functions, let us begin by reviewing a simpler notion that we are very comfortable with. Consider the following claim.

> The size of a random element from the set $\{0,1,2,3\}$ is expected to be $\frac{3}{2}$.

We all know that this is a short way of expressing the following more explicit statement.

> If an element is chosen uniformly at random from the set $\{0,1,2,3\}$, then we can expect its size to be $\frac{3}{2}$.

The term *random element* that appears in the first expression does not refer to any specific element that belongs to the set in consideration. Instead, it only indicates that a certain method for selecting an element be used. Whenever the term is used, it is followed by a claim to some expected value, and one is to understand that the probability distribution to be used in computing the claimed expected value is to be the uniform distribution.

Sometimes the specific element that has been chosen is referred to as *the* random element. This usage of the term is a source of confusion, since no randomness remains in the chosen specific element once the process of selecting from the set is complete. The randomness one might feel to be present in the chosen specific element is related to our tendency to view the number zero as being less random than a large number and is irrelevant to the situation under discussion.

Let us now turn to random functions. The following sentence presents a typical usage of the term random function.

> The image size of a random function $F : \{0,1\} \to \{0,1\}$ is expected to be $\frac{3}{2}$.

Once again, we should take this simply as a short way of expressing the following more explicit statement.

> Consider the set of all functions $F : \{0,1\} \to \{0,1\}$. If a function is chosen uniformly at random from this set of four functions, then we can expect the image size of the selected function to be $\frac{3}{2}$.

The term *random function* does not refer to a concrete function that belongs to the set of functions in consideration. Instead, the phrase specifies that a certain method of *selecting* a function be used. Once the selection has been made, no randomness remains in the resulting specific and fixed function. The term random function is always accompanied by a claim to an expected value and the term is used to indicated that the uniform distribution on the set of functions is to be used in computing the claimed expectation.

So far, there seems to be no ambiguity concerning the notion of random functions. The usage of the term as explained above is a straightforward extension of our common uses of the term random element or random number. We now turn to the following quote[3] from a textbook [11, Section 5.6], that explains random functions.

> A random function $F : \mathscr{A} \to \mathscr{B}$ is a function which assigns independent and random values $F(x) \in \mathscr{B}$ to all arguments $x \in \mathscr{A}$.

This is the *definition* of a random function that cryptographers are accustomed to. Whereas, in our previous discussion, we could not define a random function and only explain the usage of the term, the above quote seems to be defining a random function as a concrete object. In fact, at times, readers may have come across sentences that resemble the following.

> If $F : \{0,1\} \to \{0,1\}$ is *the* random function, then its image size is expected to be $\frac{3}{2}$.

[3] Only some notational changes have been made.

The random function referred to in this usage example seems to be the one that has just been defined through the textbook quote, rather than have anything to do with the previously mentioned process of selecting from a set of functions.

However, a closer look reveals that the quoted definition does not define a true function. Instead, the definition provides a process by which a specific function may be *constructed* or defined. A function that is being constructed is not a function in the set theoretic sense until it has fully been defined on all its inputs. Hence, at least to a strict mathematician, the quoted definition from the textbook does not make sense. Furthermore, as soon as the random function is fully specified, thus earning the status of a true function, no randomness can be found with the function.

Hence, once again, we are left with a usage example where the appearance of the term random function specifies certain actions to be taken in constructing a function, rather than the definition of a concrete object. It should now be noticeable that the latest usage example makes a claim to a certain expected value, as was with all our previous usage examples.

We have explained two approaches to the term random function. In both approaches, there is no set theoretic object that corresponds to the term *random function* and the appearance of the term only indicates that a certain process for specifying a function be used in computing an expected value. Sometimes the selected or fully constructed function is referred to as the random function, but no randomness remains in this specific function. The current situation is strictly analogous to the situation with random elements, except that there is more room for confusion with random functions in that we intuitively tend to view a fully defined specific function as still looking rather random, if we can not find a simple rule that can be used to specify the output corresponding to each input.

Let us distinguish between the two approaches to random functions by referring to them through the terms *selection* and *construction*. We will now show that the selection and construction approaches are equivalent. Let us first fix any specific function $F_0 : \mathscr{A} \to \mathscr{B}$. Suppose we construct a function by assigning a randomly and independently chosen element of $\mathscr{B}$ to each element of $\mathscr{A}$. Then, for the randomly constructed function to become identical to F_0, each randomly selected image point must match the corresponding output of the function F_0, and the probability for such an event to occur must be $\left(\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}$. On the other hand, if a function is selected uniformly at random from the set of functions $\mathscr{B}^{\mathscr{A}}$, since the size of $\mathscr{B}^{\mathscr{A}}$ is B^{A}, the probability for F_0 to be chosen is $\frac{1}{\mathsf{B}^{\mathsf{A}}}$. Hence, the construction approach to specifying a function brings about the uniform distribution on $\mathscr{B}^{\mathscr{A}}$. We can conclude that the selection and construction approaches to random functions are identical in that they provide the same distribution with which to compute an expected value.

Even though the selection and construction approaches result in the same distribution on $\mathscr{B}^{\mathscr{A}}$, it is often easier to work with the construction approach when explicitly computing expectations. One can roughly say that the selection of many points is equivalent to the selection of a single function and one can compute an expectation defined over the distribution of functions by suitably combining many expectations defined over the distribution of points. The expectations defined over points are simpler and easier to deal with. In our subsequent usages of the random function notion, we shall not distinguish between the two approaches unless necessary, but will mainly use expressions that can be read more naturally with the construction approach in mind.

We briefly remark that the randomly constructed function is very close to a random oracle. The only difference is that, while one constructs random functions by selecting random image points by oneself, the random oracle is treated as an external object to which one makes queries.

3.2 Random function arguments

There will be many arguments given in this paper that use the notion of random functions and not all of them will be strictly correct in the mathematical sense. In this subsection, we identify and discuss one such logical gap that is not easily noticed and then provide justification for still using such an argument.

Before discussing proofs that use the notion of random functions, we state a technical lemma that will be used throughout this paper. The proof is given in Appendix A.

Lemma 1 *For positive integers* A *and* B*, we have*

$$\left|\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right)-\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\right| < \left\{\frac{1}{2}\frac{\mathsf{A}}{\mathsf{B}^2}+\frac{1}{(\mathsf{A}+1)!}\left(\frac{\mathsf{A}}{\mathsf{B}}\right)^{\mathsf{A}+1}\right\}\exp\left(\frac{\mathsf{A}}{\mathsf{B}}\right).$$

Hence, if A *and* B *are large integers such that* $\mathsf{A}=O(\mathsf{B})$*, then*

$$\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right)\approx\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}$$

is an accurate approximation.

This lemma shows that the error in the approximation $\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\approx\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right)$ is extremely small. For example, when $\mathsf{A}=\mathsf{B}$, the bound stated in the lemma is at most $\frac{e}{\mathsf{B}}$. This approximation will be used so frequently in this paper, that we shall not even reference the above lemma when applying it.

Our first example to logical arguments that use random functions is now given.

Lemma 2 *When the finite sets* $\mathscr{A}$ *and* $\mathscr{B}$ *are sufficiently large with* $\mathsf{A}\le\mathsf{B}$*, the image size of a random function* $F:\mathscr{A}\to\mathscr{B}$ *is expected to be*

$$\mathsf{B}\left\{1-\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\right\}\approx\mathsf{B}\left\{1-\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right)\right\}.$$

Proof The approximation follows from the condition $\mathsf{A}\le\mathsf{B}$ and Lemma 1. We mention this since we are giving the very first application of the approximation, but our subsequent uses of the approximation will usually be silent.

It suffices to show the following statement, which is the interpretation of the claim in terms of the construction approach to random functions.

> If one constructs a function $F:\mathscr{A}\to\mathscr{B}$ by assigning independently and randomly chosen elements of $\mathscr{B}$ to each input element of $\mathscr{A}$, then the image size of the resulting function is expected to be $\mathsf{B}\left\{1-\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\right\}$.

Let us focus on the elements of $\mathscr{B}$ that remain as non-image points after the function construction is complete. We want to compute the expected ratio of such elements among $\mathscr{B}$.

If necessary, we can enumerate all elements of the finite set $\mathscr{A}$, so that we may refer to each element as an *i*-th element. When a random point of $\mathscr{B}$ is assigned to the first element of $\mathscr{A}$, the ratio of points among $\mathscr{B}$ that remain as non-images will become $\left(1-\frac{1}{\mathsf{B}}\right)$. After a random point of $\mathscr{B}$ has been assigned to the second element of $\mathscr{A}$, we can expect $\left(1-\frac{1}{\mathsf{B}}\right)^2$ of the points of $\mathscr{B}$ to remain as non-image points.

Since each assignment is independent of all other assignments, we may conclude that when every element of $\mathscr{A}$ has been made to map to some element of $\mathscr{B}$, the ratio of the non-image points among $\mathscr{B}$ is expected to be $\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}$. Hence, the ratio of the image points among $\mathscr{B}$ is expected to be $\left\{1-\left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\right\}$, as stated by the lemma. □

The following is an easy generalization of the above lemma.

Lemma 3 *Let $F : \mathscr{N} \to \mathscr{N}$ be the random function on a finite set of size* N. *If $\mathscr{M} \subset \mathscr{N}$ is of size m_0, then the size of $F(\mathscr{M})$ is expected to be*

$$m_1 = \mathsf{N}\left\{1 - \left(1 - \frac{1}{\mathsf{N}}\right)^{m_0}\right\}.$$

Proof It is clear that in dealing with the construction approach to random functions, the order in which elements of the domain are assigned elements of the range does not affect the expected value being computed. Hence, we can choose to give assignments to all elements of $\mathscr{M}$, before assigning elements to other points of the domain. It is also possible to stop the assignments when we are finished with $\mathscr{M}$, since the rest cannot affect the size of $F(\mathscr{M})$. With this observation, it now suffices to reread the proof of Lemma 2 with the replacements $\mathscr{A} \leftarrow \mathscr{M}$, $\mathsf{A} \leftarrow m_0$, $\mathscr{B} \leftarrow \mathscr{N}$, and $\mathsf{B} \leftarrow \mathsf{N}$. $\square$

Let us emphasize two points about this lemma. The first is that the statement of this lemma does not contain averaging over all sets of size m_0. The claim concerning the expected image size holds true for every set $\mathscr{M}$ of size m_0. The secondly point is that the value claimed by this lemma is the exact expected value and is not an approximation.

The proof given above that involved random functions contains no logical gaps. In the rest of this section, we will try to illuminate the hidden difficulty involved in proving the next lemma and also try to convince the readers that this logical gap may safely be ignored. Given a function $F : \mathscr{N} \to \mathscr{N}$, we shall write $F^k = F \circ \cdots \circ F$ for the k-times iterated composition of the function F.

Lemma 4 *Let $F : \mathscr{N} \to \mathscr{N}$ be the random function on a finite set of size* N. *Given any nonnegative $m_0 \le \mathsf{N}$, define m_k through the recursive relation*

$$\frac{m_{i+1}}{\mathsf{N}} = 1 - \exp\left(-\frac{m_i}{\mathsf{N}}\right) \qquad (i = 0, \ldots, k-1).$$

If $\mathscr{M} \subset \mathscr{N}$ is of size m_0, then the iterated image size $|F^k(\mathscr{M})|$ is expected to have the asymptotic form m_k, as N is sent to infinity.

The special case of this lemma for when the input set $\mathscr{M}$ is the complete domain $\mathscr{N}$ may be found in [5, 11]. The case when $\mathscr{M}$ is strictly smaller than the complete domain is used in [13] to state the success probability of a non-perfect rainbow table.

The statement of this lemma requires some clarification. It is not difficult to show that, if m_0 is fixed, then we have $\lim_{\mathsf{N}\to\infty} m_k = m_0$, for any k. This interpretation of the phrase asymptotic form would make this lemma correct, but would also make it completely valueless. On the other hand, we know from Lemma 3 that, for any fixed N, even the first value m_1 given by this lemma can only be an approximation and not the exact expected value. Hence, a mentioning of some kind of limiting behavior cannot be avoided if the claim is to be stated correctly. For now, let us understand the statement as claiming that the defined value m_k is the correct expected value if the error present in the approximation $\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right) \approx \left(1 - \frac{1}{\mathsf{B}}\right)^{\mathsf{A}}$ is ignored. We hope to convince the readers by the end of this section that this approximation is not the only factor that forces us to state the iterated image size in an asymptotic form.

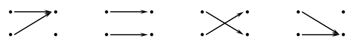
Since we know the exact image size expectation for the first iteration of F, let us see if this knowledge can be extended directly to the second iteration.

Lemma 5 (Incorrect) *Let $F : \mathscr{N} \to \mathscr{N}$ be the random function acting on a finite set of size N. If $\mathscr{M} \subset \mathscr{N}$ is of size m_0, then the size of $F^2(\mathscr{M})$ is expected to be m_2, where*

$$m_1 = \mathsf{N}\Big\{1 - \Big(1 - \frac{1}{\mathsf{N}}\Big)^{m_0}\Big\} \quad \textit{and} \quad m_2 = \mathsf{N}\Big\{1 - \Big(1 - \frac{1}{\mathsf{N}}\Big)^{m_1}\Big\}.$$

The statement of this lemma was intensionally written without an application of the approximation $\exp\big(-\frac{\mathsf{A}}{\mathsf{B}}\big) \approx \big(1 - \frac{1}{\mathsf{B}}\big)^{\mathsf{A}}$. As with the single iteration case, given by Lemma 3, we would expect this twice iterated version to hold exactly, i.e., contain no approximation. However, we can *disprove* this statement with an explicit counterexample.

The set of all functions $F : \{0,1\} \to \{0,1\}$ can be visualized as follows.



When the input set $\mathscr{M}$ is a single point, the image size expectation is clearly 1. This is in agreement with the value

$$2\Big\{1 - \Big(1 - \frac{1}{2}\Big)^1\Big\} = 1,$$

computed according to Lemma 3. When the input set is the complete domain $\{0,1\}$, the image size expectation is

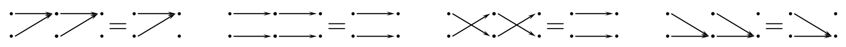
$$E_F\big[|F(\{0,1\})|\big] = \frac{1}{4}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 1 = \frac{3}{2},$$

and this is also identical to the value

$$E_F\big[|F(\{0,1\})|\big] = 2\Big\{1 - \Big(1 - \frac{1}{2}\Big)^2\Big\} = \frac{3}{2},$$

computed according to Lemma 3. Hence, as we have already proved, Lemma 3 holds exactly for the $\mathscr{N} = \{0,1\}$ case, regardless of the input set size.

Now, the four functions $F^2 = F \circ F$ can be visualized as follows.



Hence, when the input set $\mathscr{M}$ is taken to be the complete domain, the expected image size of the double iteration is

$$E_F\big[|F^2(\{0,1\})|\big] = \frac{2}{4}\cdot 1 + \frac{2}{4}\cdot 2 = \frac{3}{2}. \tag{1}$$

The corresponding value claimed by Lemma 5 is

$$2\Big\{1 - \Big(1 - \frac{1}{2}\Big)^{2\{1-(1-\frac{1}{2})^2\}}\Big\} = 2\Big\{1 - \Big(1 - \frac{1}{2}\Big)^{\frac{3}{2}}\Big\} \approx 1.293. \tag{2}$$

The two values are clearly in disagreement.

Since Lemma 3 always holds exactly, in particular, since it holds exactly in the environment of the counterexample, we can conclude that Lemma 5 does not logical follow directly from Lemma 3. In particular, we cannot hope to claim the correctness of Lemma 4, our current goal, directly from the correctness of the single step result Lemma 3, at least without providing additional arguments.

The counterexample does not rule out the possibility that Lemma 5 may still be asymptotically true as N is sent to infinity, when the word *asymptotically* is properly interpreted, but the focus of our argument here is that a statement concerning multiple iterations is not a direct consequence of the single iteration statement.

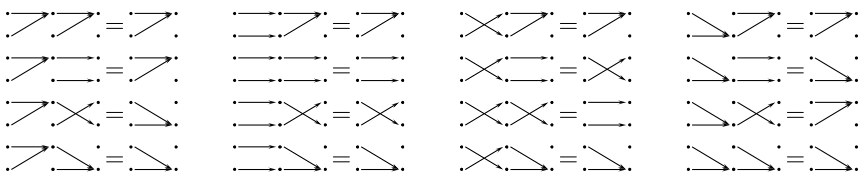
A cryptographer would naturally attempt to fix the current situation by relaxing the strict correlation between the two functions that are being composed. This is considered next.

Lemma 6 **(Incorrect)** *Let $F : \mathscr{N} \to \mathscr{N}$ and $G : \mathscr{N} \to \mathscr{N}$ be two independent random functions operating on a finite set of size N. If $\mathscr{M} \subset \mathscr{N}$ is of size m_0, then the size of $G\big(F(\mathscr{M})\big)$ is expected to be m_2, where*

$$m_1 = \mathsf{N}\Big\{1 - \Big(1 - \frac{1}{\mathsf{N}}\Big)^{m_0}\Big\} \quad \textit{and} \quad m_2 = \mathsf{N}\Big\{1 - \Big(1 - \frac{1}{\mathsf{N}}\Big)^{m_1}\Big\}.$$

This second version for the double iteration case seems structurally much simpler to analyze than our previous attempt, given as Lemma 5. One might be tempted to say that this version is a trivial consequence of the single step result, stated by Lemma 3, but, once again, we can provide a counterexample.

We return to the case of $F : \{0,1\} \to \{0,1\}$. The set of all possible double iterations of the four mappings can be visualized as follows.



When the input set $\mathscr{M}$ is the complete domain $\{0,1\}$, after separately counting the number of functions with image sizes one and two, the expected image size can be computed as

$$E_{F,G}\Big[\big|G\big(F(\{0,1\})\big)\big|\Big] = \frac{12}{16} \cdot 1 + \frac{4}{16} \cdot 2 = \frac{5}{4}. \tag{3}$$

The corresponding value, as claimed by Lemma 6, is equal to what was previously computed in (2). Since the two values disagree, it is clear that Lemma 6 cannot be true, unless it is modified into an asymptotic claim. Once again, we can conclude that Lemma 4, even when relaxed to allow independent random functions at each iteration, is not a direct logical consequence of the single iteration result Lemma 3.

We have disproved the double iteration case Lemma 5 and even the seemingly simpler Lemma 6 with explicit counterexamples. On the other hand, we know from experience that Lemma 4 works quite well in accurately predicting the behavior of iterations done with specific functions. Let us attempt to prove the incorrect Lemma 6 directly so as to locate the source of this apparent contradiction.

Let us briefly use the language pertaining to the selection approach to random functions. We start our examination of Lemma 6 by rewriting the conclusion we want to obtain as

$$E_{F,G\in\mathscr{N}^{\mathscr{N}}}\Big[\big|G\big(F(\mathscr{M})\big)\big|\Big] = \mathsf{N}\Big\{1 - \Big(1 - \frac{1}{\mathsf{N}}\Big)^{E_{F\in\mathscr{N}^{\mathscr{N}}}\big[|F(\mathscr{M})|\big]}\Big\}. \tag{4}$$

Here, the exponent on the righthand side given in terms of expectation is justified through Lemma 3. The lefthand side expectation can be written according to its definition as follows.

$$\begin{aligned} E_{F,G\in\mathscr{N}^{\mathscr{N}}}\Big[\big|G\big(F(\mathscr{M})\big)\big|\Big] &= \frac{1}{\big(\mathsf{N}^{\mathsf{N}}\big)^2} \sum_{F,G\in\mathscr{N}^{\mathscr{N}}} \big|G\big(F(\mathscr{M})\big)\big| \\ &= \frac{1}{\mathsf{N}^{\mathsf{N}}} \sum_{F\in\mathscr{N}^{\mathscr{N}}} \frac{1}{\mathsf{N}^{\mathsf{N}}} \sum_{G\in\mathscr{N}^{\mathscr{N}}} \big|G\big(F(\mathscr{M})\big)\big|. \end{aligned}$$

Now, for every fixed $F \in \mathscr{N}^{\mathscr{N}}$, the inner summation computes the image size that is expected, when a random function G is applied to the fixed input set $F(\mathscr{M})$. Hence, the above is equal to

$$\begin{aligned}\frac{1}{\mathsf{N}^{\mathsf{N}}}\sum_{F\in\mathscr{N}^{\mathscr{N}}} E_{G\in\mathscr{N}^{\mathscr{N}}}\left|G\big(F(\mathscr{M})\big)\right| &= \frac{1}{\mathsf{N}^{\mathsf{N}}}\sum_{F\in\mathscr{N}^{\mathscr{N}}} \mathsf{N}\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^{|F(\mathscr{M})|}\Big\} \\ = E_{F\in\mathscr{N}^{\mathscr{N}}}\Big[\mathsf{N}\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^{|F(\mathscr{M})|}\Big\}\Big] &= \mathsf{N}\Big\{1-E_{F\in\mathscr{N}^{\mathscr{N}}}\Big[\Big(1-\frac{1}{\mathsf{N}}\Big)^{|F(\mathscr{M})|}\Big]\Big\}.\end{aligned} \tag{5}$$

A moment of thought shows that this final form cannot be equal to the righthand side of (4). For example, given a few numbers, we know very well that the average of their inverses is not equal to the inverse of their average. The expectation operator generally does not commute with a nonlinear operator, and exponentiation with the base set to $\big(1-\frac{1}{\mathsf{N}}\big)$ is certainly a nonlinear operator. Hence, we come to the conclusion that Lemma 6 cannot be true.

Our failure to connect the end of (5) to the righthand side of (4) is due to the fact that the expectation operator does not commute with the exponentiation operator in general. However, there is one situation where the expectation operator does commute with a nonlinear operator. Namely, this is when the multiple numerical inputs being considered are all identical. For example, if we are given a few numbers, but the numbers are all the same, then the average of their inverses is trivially the inverse of their average. As an extension of this exceptional behavior, we can expect an approximate commutativity of operators when the condition of inputs being identical is slightly relaxed. That is, it is reasonable to believe that if the numerical inputs being considered are close to each other, then two values computed using the two orderings of the operators will be close to each other.

Therefore, let us consider the images of a single set $\mathscr{M}$ under different functions F and discuss the distribution of their sizes $|F(\mathscr{M})|$. A proof of the following lemma is provided in Appendix B.

Lemma 7 *Let $F : \mathscr{N} \to \mathscr{N}$ be the random function acting on a finite set of size N. For input sets $\mathscr{M} \subset \mathscr{N}$ of size m, the standard deviation $\sigma_{\mathsf{N},m}$ of the image set size $|F(\mathscr{M})|$ is asymptotically of $O(\sqrt{m})$ order.*

Recall from Lemma 3 that the expected image size under the assumptions of the above lemma is approximately $\mu_{\mathsf{N},m} = \mathsf{N}\left\{1-\exp\left(-\frac{m}{\mathsf{N}}\right)\right\}$. The condition $m \le \mathsf{N}$ and the bounds $x - \frac{1}{2}x^2 \le 1-\exp(-x) \le x$, that are valid for all $x \ge 0$, imply that $\frac{m}{2} \le \mu_{\mathsf{N},m} \le m$. Hence, the expected image size $\mu_{\mathsf{N},m}$ grows asymptotically as $\Theta(m)$.

According to Chebyshev's inequality, at least 99% of the N^{N} image sizes will fall within the range $\mu_{\mathsf{N},m} \pm 10\,\sigma_{\mathsf{N},m}$. On the other hand, the facts $\mu_{\mathsf{N},m} = \Theta(m)$ and $\sigma_{\mathsf{N},m} = O(\sqrt{m})$ imply that we will see $\frac{\sigma_{\mathsf{N},m}}{\mu_{\mathsf{N},m}}$ approaching zero as m is sent to infinity. Thus, the distribution or gathering of image sizes around the expected value $\mu_{\mathsf{N},m}$ will tighten, at least in comparison to the expected value, as m is increased. This observation can be restated in more plain terms as follows. Suppose we take some input set and measure its image size under a single randomly chosen explicit function and take it to be an estimate of the true average image size. It must be emphasized that the averaging over multiple measurements corresponding to different functions is not being performed here. In such a situation, we can expect each measurement to return a larger number *significant digits* as the scale of the experiment is increased.

It should be helpful to work briefly with some explicit numbers at this point. For parameters $\mathsf{N} = 2^{64}$ and $m = 2^{50}$ the average image sizes can be computed to be $\mu_{\mathsf{N},m} \approx 2^{49.99996}$.

For the same parameters, using details found in Appendix B, we can compute that the standard deviation of image sizes to be $\sigma_{\mathsf{N},m} \approx 2^{17.49993}$. Chebyshev's inequality insures that at least 75% of the N^{N} image sizes will lie in the range $\mu_{\mathsf{N},m} \pm 2\sigma_{\mathsf{N},m}$. In the current situation, this means that at least 75% of the image sizes appear within the range $2^{49.99996} \pm 2^{18.49993}$. Similarly, 93.75% of the image sizes lie in the range $\mu_{\mathsf{N},m} \pm 4\sigma_{\mathsf{N},m} = 2^{49.99996} \pm 2^{19.49993}$. For any practical purposes, we can believe that approximately 30 significant bits from any single measurement are highly likely to be identical to those of the expected value. These explicit numbers allow us to feel how tightly the image sizes are distributed around the expected value for large input sets.

Let us return to the subject of image size expectation under the iteration of two independent random functions. We had discussed how this was related to the approximate commutativity of two operators. The numerical inputs to the expectation and nonlinear operators that need to be considered are the multiple image sizes $|F(\mathscr{M})|$, where the function F is made to run over all elements of $\mathscr{N}^{\mathscr{N}}$. When $|\mathscr{M}|$ is large, these image sizes will be such that a vast majority of them will have many significant digits identical to the significant digits of the average value. Hence, we are somewhat justified in computing the expected image size of $\{G(F(\mathscr{M}))\}_{F,G}$ as if the random function G was simply given an input set of size $E_F\big[|F(\mathscr{M})|\big]$.

Although Lemma 6 is not strictly true, we can believe it to be approximately true when m_0 is large. Here, being approximately true should be understood in the sense that we can expect the formulae to provide many correct significant digits. It should not be taken to imply that the absolute error size, which would grow with m_0, is small.

We now have fully discussed the iterated image size under two independent random functions. Recall that our current goal is to justify Lemma 4, which requires the two iterated functions to be identical. Since we already know that it cannot be inferred directly from the truth of Lemma 3, the single step statement, we turn to providing a plausible justification for its use in practice.

The image sizes under a single application of the random function were already seen to be tightly clustered, when a large input set is in use. As for inputs sets that are small in size compared to $\mathscr{N}$, recall that $\lim_{\mathsf{N}\to\infty} m_1 = m_0$, when m_0 is fixed. This implies that the clustering of image sizes is trivially true for small input sets. Let us take this belief to the extreme and make the unrealistic assumption that the relation of Lemma 3, i.e.,

$$\text{image size} = \mathsf{N}\left\{1 - \left(1 - \frac{1}{\mathsf{N}}\right)^{\text{input size}}\right\}, \tag{6}$$

concerning the input and image sizes of a single iteration, holds exactly true, not just on average, but for every choice of the function F and input set $\mathscr{M}$. Under this unrealistic assumption, the iterative relation of Lemma 4 would be true for each function F, over any number of iterations. This automatically implies that the iterative relations are true when averaged over all functions. In this argument, we are not requiring the functions that are being composed to be independent from each other.

The use of Lemma 4 as an approximation will now be justified, if relaxing our unrealistic assumption does not cause trouble. We first consider the discouraging extreme situations. When the function F is bijective, so that (6) is far from accurate for F, it is clear that this error will persist throughout all iterations. That is, the error is not dampened through the multiple iterations. At the other extreme is the case when the function F is the constant function. The situation given by the iteration of this function is not reflected by (6) in any way. Hence, the the iterative relations appearing in Lemma 4 does not hold true for these individual functions that are extremely far away from our unrealistic assumption. Nevertheless, it should be noted that the number of these extreme functions is very small compared to the number of all functions.

In reality, for any explicit function, relation (6) can not be exactly true for every input set $\mathscr{M}$. Instead, it will be such that (6) slightly overestimates the image size for some input sets and also slightly underestimates for other input sets. Now, suppose that for some F and $\mathscr{M}$, the size of $F(\mathscr{M})$ is somewhat smaller than what is estimated by (6). Note that it is unlikely for the points of the image set $F(\mathscr{M})$ to be somehow related to the points of $\mathscr{M}$. In other words, there is no reason to believe that an application of F to the image set $F(\mathscr{M})$ will produce a second image that is also smaller than what would be estimated by (6), on a set of size $|F(\mathscr{M})|$. This independence between a subset of $\mathscr{N}$ viewed as an image set and the same subset viewed as an input set to the next iteration assures us that a sequential buildup of any local abnormal behavior is unlikely to happen. Hence, even if (6) is no longer strictly true but only approximately true for inputs to a certain specific function, the relations of Lemma 4 will remain approximately true for the same specific function.

We have shown that the validity of a statement concerning a single step of a random function does not directly imply the validity of the same statement under multiple iterations. There are works, especially those concerning time memory tradeoffs, that ignore the logical gap that we have discussed. These arguments are typically written in the language of classical occupancy problems, but are in fact random function arguments that ignore behavior under iterations. In the opposite direction, we have argued that results for a single iteration of a random function may be generalized to multiple iterations when the space that is being dealt with is very large and the statements are understood to be approximations.

The intension of this subsection was not in testing the validity of Lemma 4. In fact, although the authors of the current paper are unfit to verify its correctness, a full proof is provided in [5] for the special case when $\mathscr{M}$ is the full domain. When the validity of Lemma 4 is taken for granted, the long discussions of this section make the following claim quite plausible.

Lemma 8 (Belief) *Let $F : \mathscr{N} \to \mathscr{N}$ be any explicitly function acting on a finite set of size N which is free of any structure that one may feel as affecting the iterated image sizes in a certain way. If $\mathscr{M} \subset \mathscr{N}$ is of size m_0, then the size of the iterated image $F^k(\mathscr{M})$ will be approximately m_k, where m_k is defined recursively through*

$$\frac{m_{i+1}}{\mathsf{N}} = 1 - \exp\Big(-\frac{m_i}{\mathsf{N}}\Big) \qquad (i = 0, \ldots, k-1).$$

In short, we are claiming that Lemma 4 is likely to hold true quite accurately, in the sense that many significant digits are provided, for most functions and not just on average. We emphasize that this is a *belief*, which we shall freely use in this paper. In fact, the first sentence of this lemma is ambiguous and we cannot even hope for a mathematical proof. Lemma 4 is a statement concerning an average behavior and it cannot imply anything about the behavior of each individual function, even if we had a strictly logical proof.

The identity function and the constant function certainly are not the subject of the above lemma. One notable function that is not covered by the above lemma is the r-adding walk iteration function [14], which is used by certain algorithms for solving discrete logarithm problems. The rho-length associated with this function tends to be slightly longer than that expected of a random function, especially when a small r is used. This indicates that its iterated image sizes will be larger than those expected of a random function.

Recall that much of the discussion in this subsection was about whether we may interchange the order of application of the expectation operator and a nonlinear operator. We now have reasons to believe that the order of application may be interchanged without introducing big errors when the space being dealt with is large. Throughout this paper, our *random function arguments* will carelessly disregard the orders in which the operators are applied.

4 Applying Time Memory Tradeoff to Password Hashes

One usually states the objective of a tradeoff algorithm as the inversion of a one-way function. A closer look reveals that there are two versions of the inversion problem and we will explain how one of these corresponds to the application of tradeoff technique on password systems. Issues concerning the use of random functions in the theoretic analysis of tradeoff algorithms are also discussed in this section.

4.1 Password hash

Let us briefly explain how the security features of many file formats that rely on passwords for access control work in its very basic form.

The designer of the system chooses and fixes a one-way function H. This one-way function is a part of the file format specification and is usually considered to be public. In fact, the one-way function definition can be extracted from the related software even if it was not originally made public. When the owner of a file following this format wants access control to be applied to the file, the user supplies a password $\mathbf{x}$. An encryption key is derived from the password, and the main content of the file is replaced by its encryption under this key. Then the image $\mathbf{y} = H(\mathbf{x})$ of the user password, under the one-way function specified for the file format, is recorded in the file. Finally, any record of the encryption key and the raw password supplied by the user is destroyed.

Later, when authentication is required for file access, the supporting software asks for a password. The one-way function image $H(\mathbf{x}')$ of the newly supplied password $\mathbf{x}'$ is computed by the software and is compared with the corresponding information $\mathbf{y}$ stored within the file. If a perfect match $\mathbf{y} = H(\mathbf{x}')$ is found, the main body of the file is decrypted using the key derived from the password and access to the decrypted content is granted. Note that the one-way function image $\mathbf{y}$ of the correct password is stored within the file without any protection and is accessible to anyone that has obtained the file.

User authentication procedure for system login works in much the same way. At the time of initial user registration to the system, the one-way function image of the password supplied by the user is recorded in a file that is stored within the system. In this case, access to the one-way function images may be harder for the attacker than the above case, but this information is often sent over the network in the clear to a group of computers, so that each of these computers may allow authenticated logins to a user that has registered at a central server.

As we have stated earlier, we shall refer to the one-way function image as the *password hash* and the input as the *password*, regardless of whether the one-way function to be attacked through the tradeoff technique is related to a password authentication system.

4.2 Uniqueness of the pre-image to a password hash

Our first question is whether a password hash uniquely determines the password.

Proposition 9 *Let $H : \mathscr{P} \to \mathscr{H}$ be a random function. Given any $\mathbf{x} \in \mathscr{P}$, the number of inputs that H maps to $H(\mathbf{x})$ is expected to be $1 + \frac{|\mathscr{P}|-1}{|\mathscr{H}|}$.*

Proof When using the construction approach to a random function, we are free to choose the order in which function value assignments are made to each domain element. So let us first assign a randomly chosen value of $\mathscr{H}$ to $H(\mathbf{x})$ and then define all the other function values.

The probability for any one of the later assignments to strike $H(\mathbf{x})$, which is an explicitly fixed value in $\mathscr{P}$, is $\frac{1}{|\mathscr{H}|}$. Each later assignment is independent of all other assignments, and we can expect the number of later assignments to $H(\mathbf{x})$ to be $\frac{|\mathscr{P}|-1}{|\mathscr{H}|}$. □

Readers should not misinterpret the above proposition as giving the pre-image size of a random $\mathbf{y} \in \mathscr{H}$ under a random H. For a random function H, the distribution on $\mathscr{H}$ produced by $H(\mathbf{x})$ is the uniform distribution, and every $\mathbf{y} \in \mathscr{H}$ is expected to have $\frac{|\mathscr{P}|}{|\mathscr{H}|}$-many pre-images, rather than $1 + \frac{|\mathscr{P}|-1}{|\mathscr{H}|}$. This is not in contradiction with the proposition, as the proposition deals with the distribution on $\mathscr{H}$ produced from random inputs by the specific H that has been constructed, and this is different from the uniform distribution on $\mathscr{H}$. Those points of $\mathscr{H}$ that lie outside $H(\mathscr{P})$, for the specifically constructed H, do not have any chance of appearing.

One can also ask for the pre-image size of a random password hash $\mathbf{y} \in H(\mathscr{P})$. Note that this question can only be asked after the random function H has fully been constructed. The corresponding answer will depend on the size of $H(\mathscr{P})$, but should be close to

$$\frac{|\mathscr{P}|}{E(|H(\mathscr{P})|)} \approx \frac{1}{1-\frac{1}{e}} \approx 1.582,$$

when $|\mathscr{P}| = |\mathscr{H}|$. Once again, this question is not related to the content of the above proposition. The current question deals with the uniform distribution on $H(\mathscr{P})$, which is different from the distribution on $H(\mathscr{P})$ given by the fully specified H. Those points with larger pre-image sets will have a larger probability of appearing than those with smaller pre-image sets.

Consider an application of the tradeoff technique to a blockcipher whose key length is identical to the block length. In such a case, one is working with $|\mathscr{P}| = |\mathscr{H}|$ and Proposition 9 states that there will be approximately two keys, on average, that map to a given target ciphertext. This is probably larger than what many would have naively expected. Of course, in practice, one usually assumes the use of a second ciphertext to almost uniquely identify the key. In fact, if one interprets the key to two-ciphertexts mapping as a new one-way function, then Proposition 9 claims that the key is almost uniquely determined from the two ciphertexts.

Let us next discuss what Proposition 9 implies for systems that rely on passwords for access control. These systems are usually designed so that the space $\mathscr{H}$ of potential hash values is significantly larger than the space $\mathscr{P}$ of admissible passwords. A typical password hash would be a bit string of at least 128 bits in length and the number of alphanumeric passwords consisting of ten characters is only $62^{10} \approx 2^{59.5}$. In such a case, Proposition 9 shows that a password hash $H(\mathbf{x})$, produced from a password $\mathbf{x}$, will almost always identify $\mathbf{x}$ uniquely.

Furthermore, in practice, the set of all passwords admissible by the security system is not very important. Since passwords chosen by humans are not uniformly distributed within the complete admissible password space, the tradeoff attacker first fixes a manageable subset

$\mathscr{P}' \subset \mathscr{P}$ from the set of all passwords and decides to be satisfied with recovering only those passwords that lie in $\mathscr{P}'$. The size of this subset is determined by the computational power that the attacker can allocate to the pre-computation phase and should preferably cover the passwords that are most likely to be used. Under such a setting the password hash set $\mathscr{H}$ is immensely larger than the set of passwords $\mathscr{P}'$ that is being considered and hence the password hash determines the password uniquely.

For the remainder of this paper, we assume that the target system for the application of the tradeoff technique is such that $|\mathscr{P}| \ll |\mathscr{H}|$, implying that the password hash uniquely determines the password.

4.3 The reduction function

The tradeoff technique requires the one-way function to be iterated. Since the range of the one-way function is usually larger than the domain, iteration is achieved by utilizing a *reduction function* $R : \mathscr{H} \to \mathscr{P}$. The role of the reduction function is to let a password hash be interpreted as another password. As any theoretic treatment of the tradeoff technique assumes $R \circ H$ to be a random function, let us check whether this is appropriate.

Proposition 10 *Let $|\mathscr{P}|$ be a divisor of $|\mathscr{H}|$, so that $\frac{|\mathscr{H}|}{|\mathscr{P}|}$ is an integer. Let $R : \mathscr{H} \to \mathscr{P}$ be any fixed function that is pre-image uniform in the sense that it is exactly $\frac{|\mathscr{H}|}{|\mathscr{P}|}$-to-1. If $H : \mathscr{P} \to \mathscr{H}$ is a random function, then $R \circ H : \mathscr{P} \to \mathscr{P}$ is a random function.*

Proof In more precise terms, we want to show that the distribution on $\mathscr{P}^{\mathscr{P}}$, produced from the uniform distribution on $\mathscr{H}^{\mathscr{P}}$, through the mapping $H \mapsto R \circ H$, is the uniform distribution.

Let $F_0 : \mathscr{P} \to \mathscr{P}$ be any specific function. It suffices to show that, after random construction of a function $H : \mathscr{P} \to \mathscr{H}$, we will find $R \circ H = F_0$ with probability $\frac{1}{|\mathscr{P}|^{|\mathscr{P}|}}$. It is clear that $\{R^{-1}(\mathbf{z})\}_{\mathbf{z} \in \mathscr{P}}$ is a partition of $\mathscr{H}$ into cells of size $\frac{|\mathscr{H}|}{|\mathscr{P}|}$. The event $F_0 = R \circ H$ will happen if and only if the value assigned as $H(\mathbf{x})$ belongs to the cell $R^{-1}\big(F_0(\mathbf{x})\big)$, for every $\mathbf{x} \in \mathscr{P}$. Since the size of $R^{-1}\big(F_0(\mathbf{x})\big)$ is always $\frac{|\mathscr{H}|}{|\mathscr{P}|}$, and since the assignment to $H(\mathbf{x})$ is independent and random for every x, the probability of arriving at $F_0 = R \circ H$ is

$$\Big(\frac{|\mathscr{H}|/|\mathscr{P}|}{|\mathscr{H}|}\Big)^{|\mathscr{P}|} = \frac{1}{|\mathscr{P}|^{|\mathscr{P}|}},$$

as claimed. □

Every application of the time memory tradeoff technique on a security system involves a specific one-way function $H : \mathscr{P} \to \mathscr{H}$ and there is no strictly logical reason to believe that the specific H will display the properties expected of a random function. Hence we need to discuss if predicting the behavior of an explicit tradeoff implementation with arguments concerning random functions can be justified in practice.

There can be two ways to resolve this problem. The first is to appeal to our intuition. When one ignores his knowledge of the inner working of the given specific function, it will seem as if the function is returning independently and randomly generated values to each given input. Hence, viewed from the outside, it looks as if the specific function is the random function in the construction sense. The second argument, which seems slightly more plausible, is that the one-way function used in the security system is in fact a function that has been selected from the pool of all functions. As we discussed earlier in Section 3.2, when the spaces involved are sufficiently large, unless we had chosen the one-way function in an unusual way, any property exhibited by a specific function will be close to the property averaged over all functions.

We have thus partly justified the use of random functions in place of specific one-way functions $H : \mathscr{P} \to \mathscr{H}$ when analyzing the behavior of time memory tradeoffs. What we have shown through Proposition 10 is that if we may treat the specific one-way function H as a random function, then the same can be done with the function $R \circ H : \mathscr{P} \to \mathscr{P}$. Hence, throughout this paper, while analyzing the behavior of time memory tradeoffs, we shall work with a random function $F : \mathscr{N} \to \mathscr{N}$.

The discussion of reduction functions now brings us to another logical gap that frequently appears in random function arguments, that is specific to the analysis of time memory tradeoffs. It is often the case that multiple tables are used in a tradeoff implementation and any analysis of tradeoff properties will assume these tables to be independent. Although a different reduction function or a family of reduction functions is used with each table, it is not true that the tables are independent. In the language of the construction approach to random functions, assignments made during computation of an earlier table will prevent later assignments to be made independently.

Suppose we try to analysis a time memory tradeoff properties that involve multiple tables under our simple assumption that $F : \mathscr{N} \to \mathscr{N}$ is a random function. Then this will require averaging over functions of a value that combines multiple figures that come from different tables and these multiple figures will be correlated to each other. This will be very complex and difficult to handle. Once again, since the domain and range spaces that are being considered are usually very large, we assume the interchange of the expectation operator and any nonlinear operator is possible and present analysis that averages over functions before combining the figures to arrive at the final expectation. This is equivalent to assuming that multiple independently constructed random functions were used to create different tables.

4.4 Two versions of the inversion problem

We have already mentioned that we shall work in the situation $H : \mathscr{P} \to \mathscr{H}$ where the sets satisfy $|\mathscr{P}| \ll |\mathscr{H}|$, so that a password hash almost always determines a unique password. We also know that any analysis of time memory tradeoff behavior is done with a random function $F : \mathscr{N} \to \mathscr{N}$, whose image does not uniquely determine the input. The two functions are related through the correspondence $H \mapsto F = R \circ H$.

Given $\mathbf{y} = H(\mathbf{x})$, the unique password $\mathbf{x}$ is obtained through the tradeoff algorithm, which uses F, as follows. The tradeoff algorithm is given $R(\mathbf{y})$ to process and returns inputs $x \in \mathscr{P}$, such that $F(x) = R(\mathbf{y})$. This relation may be restated as $R(H(x)) = R(H(\mathbf{x}))$ and does not necessarily imply $x = \mathbf{x}$. Hence one tests whether the candidate password x satisfies $H(x) = H(\mathbf{x})$, outside of the normal tradeoff algorithm. Since an output of H uniquely determines the input, fulfillment of this test implies $x = \mathbf{x}$ and recovery of the correct password.

As discussed in the previous subsection, the number of inputs to F satisfying $F(x) = R(\mathbf{y})$ will only be two on average, and the number of such tests done outside the tradeoff algorithm will be very small. Hence, the cost of such tests may be ignored during complexity analysis.

During a tradeoff algorithm analysis, one usually does not mention anything about H or R, the source of the inversion problem, and simply assume the inversion target $\mathbf{y} = F(\mathbf{x})$ is given, for some function $F : \mathscr{N} \to \mathscr{N}$. In this work, the goal of the tradeoff algorithm will be to find *the* correct password $\mathbf{x}$, rather than *any* password, corresponding to the given $\mathbf{y}$. The *any* version may be useful when working to find the pre-image of a true hash function, but the *the* version is suitable when looking for the correct password to an access control mechanism.

Since it is logically impossible to distinguish between the many pre-images with only the $\mathbf{y}$ information, our analysis will focus on whether $\mathbf{x}$ is among the possibly multiple F-pre-images to $\mathbf{y}$, returned by the tradeoff algorithm. The determination of whether each returned value is *the* correct password is assumed to be done outside the tradeoff algorithm.

The small difference of looking for *the* pre-image rather than *any* pre-image implies that the tradeoff algorithm will succeed under different circumstances. The *the* version succeeds if and only if the correct password $\mathbf{x}$ had appeared as an *input* to the one-way function F during the pre-computation phase, i.e., if $\mathbf{x}$ is among the pre-computation matrix entries excluding the ending points. On the other hand, the *any* version succeeds if and only if the image $\mathbf{y} = F(\mathbf{x})$ had appeared as the function *output* during the pre-computation phase, i.e., if $\mathbf{y}$ is among the pre-computation matrix entries excluding the starting points. The two approaches will show differences in properties such as success probability and online running time. Still, it should not be too difficult to tweak all the *the* version results given in this paper to work for the *any* version.

5 DP Tradeoff

An analysis of the DP tradeoff will be given in this section. We shall present a formula for computing the probability of success for the algorithm and also provide a tradeoff curve which takes the effects of false alarms into account. We also discuss the number of bits required to efficiently store the starting point and ending point pairs.

If a chain is generated with a random function, with the chain length bound set to $\hat{t}$, the probability of not obtaining a DP chain will be $\left(1 - \frac{1}{t}\right)^{\hat{t}} \approx e^{-\hat{t}/t}$. Rather than successively generating more chains until we have m chains, we shall generate each table from

$$m_0 = \frac{m}{1 - e^{-\hat{t}/t}} \tag{7}$$

distinct starting points. Then we can expect to collect m chains that terminate at DPs. In the limiting $\hat{t} \gg t$ case, which is of more practical interest, the two approaches will almost be the same.

All of our tradeoff algorithm analyses are done under the assumption that the one-way function is the random function. In particular, many expectations mentioned hereinafter are to be understood as those computed over the choice of random functions.

5.1 Probability of success

Let us discuss how to compute the probability of success for a DP tradeoff under a given set of parameters. We shall first present general formulas connecting pre-computation and probability of success and then show how to compute these for specific parameter. Our first lemma is quite trivial.

Lemma 11 *The number of one-way function invocations required in either creating a DP chain or stopping at the $\hat{t}$-th iteration without having reached a DP is expected to be*

$$t\,(1 - e^{-\hat{t}/t}).$$

Proof It suffices to add the probabilities of having to compute the successive iterations. Since the next iteration is computed if and only if a DP has not yet been reached, the expected one-way function invocation count is

$$\sum_{i=1}^{\hat{t}} \Big(1 - \frac{1}{t}\Big)^{i-1} = t\Big\{1 - \Big(1 - \frac{1}{t}\Big)^{\hat{t}}\Big\},$$

which we can approximate to $t\big\{1 - \exp\big(-\frac{\hat{t}}{t}\big)\big\}$. □

In the above proof, we have implicitly assumed the one-way function to be a random function and computed the probability for the first i assignments to be non-DPs. A more exact analysis would additionally consider the possibility for the next assignment to produce a previously assigned value. We have not done so as the above was good enough as an approximation.

Clearly, the success rate of a tradeoff algorithm is intimately connected to the amount of pre-computation. So, let us present a way to write down the pre-computation.

Proposition 12 *The pre-computation phase of the DP tradeoff is expected to require $\mathtt{D}_{pc}\mathtt{N}$ one-way function invocations, where the pre-computation coefficient is*

$$\mathtt{D}_{pc} = \mathtt{D}_{msr}\mathtt{D}_{nt}.$$

Proof We know from Lemma 11 that each attempt at a DP chain creation is expected to require $t(1 - e^{-\hat{t}/t})$ one-way function invocations. On the other hand, the creation of a single DP table starts with $m_0 = \frac{m}{1-e^{-\hat{t}/t}}$ chains. Together, this implies that the creation of a single DP table is expected to consume mt one-way function invocations, regardless of the $\hat{t}$ value. Hence the total pre-computation requirement may be written as $mtl = mt^2\frac{l}{t} = \mathtt{D}_{msr}\mathtt{D}_{nt}\mathtt{N}$. □

Note that the pre-computation cost of the DP tradeoff does not depend on the chain length bound $\hat{t}$.

The *coverage rate* $\mathtt{D}_{cr}$ of a DP table, is defined to be the expected number of distinct nodes that appear among the DP chains as inputs to the one-way function, divided by mt. Since we are taking starting points to be non-DPs, all of the nodes that are counted will be non-DPs. Note that the mentioned expectation is an average over the choice of one-way functions, in addition to the choice of starting points. In other words, the coverage rate is a certain expected value for the random function. Our next statement reduces the search for success rate to the computation of the coverage rate.

Proposition 13 *The success probability of the DP tradeoff is*

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

Proof If we are given $\mathbf{y} = F(\mathbf{x})$ as the inversion target, the DP tradeoff will succeed in recovering the correct password $\mathbf{x}$, if and only if $\mathbf{x}$ had appeared as one of the inputs to the one-way function during the creation of the DP table. As discussed earlier, this is not equivalent to asking for the appearance of $\mathbf{y}$ among the output values. The objective of recovering *the correct*, rather than *any* inverse, corresponds to finding $\mathbf{x}$ among the one-way function inputs.

By definition of the coverage rate, a single DP matrix is expected to contain $\mathtt{D}_{cr}mt$ distinct nodes that were used as inputs to the one-way function. Hence the processing of a single table will fail in returning the correct password with probability $\left(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\right)$. The success probability of the complete DP tradeoff process is given by

$$\mathtt{D}_{ps} = 1-\left(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\right)^{l} \approx 1-\exp\left(-\mathtt{D}_{cr}\frac{mtl}{\mathsf{N}}\right) = 1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}.$$

As discussed at the end of Section 4.3, we confide that our treatment here of separate tables as being independent is not strictly correct. □

If the storage capability of the creator of the inversion target $\mathbf{y}=F(\mathbf{x})$ is sufficiently large, it may be possible for him to compute and collect the complete set of one-way function inputs used during the DP table creation, and choose a password $\mathbf{x}$ that does not belong to this set. The DP tradeoff will always fail under such challenges. The above proof cannot capture this situation since the one-way function F was taken to be a random function while defining $\mathtt{D}_{cr}$.

In practice, this implies that, for our analysis to be correct, the hidden password $\mathbf{x}$ needs to be chosen without reference to the DP tradeoff table. Note that this is not as strong a requirement as asking for the choice of $\mathbf{x}$ to be random. The choice only needs to be unrelated to the structure of the DP matrices.

Within this subsection, all chains belonging to the DP matrix will be seen as having been aligned at the starting points, rather than at the ending points, and the starting point column will be referred to as the 0-th column.

The above expression for probability of success can only be put to use if we know how to compute the coverage rate. Our computation of the coverage rate will be done in two steps. Of the m_0 chains generated, only m will be DP chains, but we disregard this in the first step and count the number of new nodes added by each column of the complete matrix. Sum of these values gives us the total number of all distinct input entries. In the second step, we will count the number of nodes that belonged to chains not ending at DPs and subtract these from the total count.

Let us write m_j for the number of new nodes added by the j-th column. The m_0 value, stated by (7), conforms to this notation.

Lemma 14 *The number of new nodes added by each column satisfies the recurrence relation*

$$m_j = \mathsf{N}\Big\{1-\exp\Big(-\frac{m_{j-1}}{\mathsf{N}}\Big)\Big\}\Big(1-\frac{1}{t}\Big)\Big\{1-\frac{\sum_{i=0}^{j-1} m_i}{\mathsf{N}(1-1/t)}\Big\}.$$

Proof Suppose a node positioned in the $(j-1)$-th column is old, in the sense that it has appeared in one of the 0-th through $(j-2)$-th columns. Application of random function to this node will not result in a random element of $\mathscr{N}$, but a node that had appeared in one of the 1-st through $(j-1)$-th columns. Hence when counting new nodes of the j-th column we need only consider the nodes of the j-th column that are assigned as images to new nodes of the $(j-1)$-th column. Recalling Lemma 3, we write this as the $\mathsf{N}\big\{1-\exp\big(-\frac{m_{j-1}}{\mathsf{N}}\big)\big\}$ part appearing in the claimed equation.

Of the distinct entries that have appeared in the j-th column, that are not automatically old, we want to filter out the DPs. The previous count is restricted to the non-DPs by multiplying the $\left(1-\frac{1}{t}\right)$ factor.

Still, not all of these non-DPs are new nodes. Those that have appeared in previous columns are removed by multiplying $\left\{1-\frac{\sum_i m_i}{\mathsf{N}(1-1/t)}\right\}$. Notice that we have $\mathsf{N}(1-\frac{1}{t})$, rather than N, in the denominator, as we are dealing only with non-DPs at this point. □

The next two lemmas are technical computation results. We first turn the recursive formula for m_j into a difference equation concerning a certain sum of m_j.

Lemma 15 *Let $\mu_i = \frac{m_i}{\mathsf{N}(1-1/t)}$ and $\sigma_j = \sum_{i=0}^{j-1} \mu_i$. Then, σ_j satisfies the recursive formula*

$$\sigma_{j+1} - \sigma_j = \frac{m_0}{\mathsf{N}} - \frac{1}{t}\sigma_j - \frac{1}{2}\sigma_j^2 \quad \textit{with} \quad \sigma_0 = 0,$$

which is accurate up to modulo $O\left(\frac{1}{t^3}\right)$.

Proof It is straightforward to rewrite the recursive formula of Lemma 14 in terms of the notation μ_j.

$$\mu_j = \left\{1 - \exp\left(-\left(1-\frac{1}{t}\right)\mu_{j-1}\right)\right\}\left(1-\sum_{i=0}^{j-1}\mu_i\right).$$

This may be rewritten once again as

$$\exp\left(-\left(1-\frac{1}{t}\right)\mu_{j-1}\right) = 1 - \frac{\mu_j}{1-\sigma_j} = \frac{1-\sigma_{j+1}}{1-\sigma_j}.$$

Now, by taking products of both sides over $j = 1,\dots,k$, we can find

$$\exp\left(-\left(1-\frac{1}{t}\right)\sigma_k\right) = \frac{1-\sigma_{k+1}}{1-\sigma_1}.$$

We have thus arrived at a relation involving only the σ_k notation.

By expanding the exponential function in its Taylor series, we obtain

$$\sigma_{k+1} = 1-(1-\sigma_1)\left\{1-\left(1-\frac{1}{t}\right)\sigma_k + \frac{1}{2}\left(1-\frac{1}{t}\right)^2\sigma_k^2 - \cdots\right\},$$

and we can modify the above into the difference equation

$$\sigma_{k+1} - \sigma_k = \sigma_1 - \left(\sigma_1 + \frac{1}{t} - \frac{\sigma_1}{t}\right)\sigma_k - \frac{1}{2}(1-\sigma_1)\left(1-\frac{1}{t}\right)^2\sigma_k^2 + \cdots.$$

Noting that the lefthand side $\sigma_{k+1} - \sigma_k = \mu_k$ is of order $O\left(\frac{m}{\mathsf{N}}\right) = O\left(\frac{1}{t^2}\right)$, we remove every term on the righthand side of $O\left(\frac{1}{t^3}\right)$ order. This may easily be done after noting that $\sigma_1 = \mu_0$ is $O\left(\frac{1}{t^2}\right)$ and that σ_k is $O\left(\frac{mk}{\mathsf{N}}\right)$, which is at most $O\left(\frac{1}{t}\right)$. The simplified equation is now

$$\sigma_{k+1} - \sigma_k = \mu_0 - \frac{1}{t}\sigma_k - \frac{1}{2}\sigma_k^2 + O\left(\frac{1}{t^3}\right).$$

It is clear that the initial condition $\sigma_1 = \mu_0$ may be replaced by $\sigma_0 = 0$, under this recursive formula. As a final tweak, we subtract $\frac{m_0}{\mathsf{N}(t-1)}$, which is of $O\left(\frac{1}{t^3}\right)$ order, from the constant term $\mu_0 = \frac{m_0}{\mathsf{N}(1-1/t)} = \frac{m_0}{\mathsf{N}}\left(1+\frac{1}{t-1}\right)$, to arrive at the claimed formula. □

Now that we have a difference equation, we can obtain σ_k through an application of the Euler method.

Lemma 16 *For each non-negative integer k, we have*

$$m_k \approx \mathsf{N}\big(\sigma(k+1) - \sigma(k)\big)$$

where

$$\sigma(k) = \frac{\Xi^2 - 1}{t} \, \frac{\exp\left(\Xi \frac{k}{t}\right) - 1}{(\Xi+1)\exp\left(\Xi \frac{k}{t}\right) + (\Xi - 1)} \quad \textit{with} \quad \Xi = \sqrt{1 + \frac{2\mathtt{D}_{msr}}{1 - e^{-\hat{t}/t}}}.$$

Proof Let function $\sigma : \mathbf{R} \to \mathbf{R}$ be the unique solution to the differential equation

$$\frac{d}{dk}\sigma = \frac{m_0}{\mathsf{N}} - \frac{1}{t}\sigma - \frac{1}{2}\sigma^2 \quad \text{and} \quad \sigma(0) = 0. \tag{8}$$

If one defines the sequence $\{\sigma_k\}_{k\geq 0}$ through the corresponding difference equation

$$\sigma_{k+1} - \sigma_k = \frac{m_0}{\mathsf{N}} - \frac{1}{t}\sigma_k - \frac{1}{2}\sigma_k^2 \quad \text{and} \quad \sigma_0 = 0, \tag{9}$$

the the Euler method tells us that $\sigma(k)$, the evaluation of the function σ at the non-negative integer k, may be approximated by the sequence value σ_k. We may turn this the other way around to present approximate values of σ_k through the function evaluations $\sigma(k)$.

The unique solution to differential equation (8) is

$$\sigma(k) = \frac{2m_0 t}{\mathsf{N}} \frac{\exp\left(\sqrt{1 + \frac{2m_0t^2}{\mathsf{N}}}\,\frac{k}{t}\right) - 1}{\left(\sqrt{1 + \frac{2m_0t^2}{\mathsf{N}}} + 1\right)\exp\left(\sqrt{1 + \frac{2m_0t^2}{\mathsf{N}}}\,\frac{k}{t}\right) + \left(\sqrt{1 + \frac{2m_0t^2}{\mathsf{N}}} - 1\right)}.$$

The form of $\sigma(k)$ stated by this lemma is obtained when $m_0 = \frac{m}{1-e^{-\hat{t}/t}}$ and $mt^2 = \mathtt{D}_{msr}\mathsf{N}$ are substituted.

Since the definition of σ_k given by (9) is identical to the approximate recursive relation of Lemma 15, we have

$$\sigma(k) \approx \sigma_k \approx \sum_{i=0}^{k-1} \mu_i, \quad \text{where} \quad \mu_i = \frac{m_i}{\mathsf{N}(1 - 1/t)}.$$

This allows us to write

$$m_k \approx \mathsf{N}\left(1 - \frac{1}{t}\right)\big(\sigma(k+1) - \sigma(k)\big) \approx \mathsf{N}\big(\sigma(k+1) - \sigma(k)\big),$$

where the $\frac{1}{t}$ term removal is justifiable, as it is of strictly smaller order. □

This completes the first step of the coverage rate computation. The coverage rate corresponds to the number of distinct non-DP nodes contained in just the DP chains, but the currently computed m_k includes all points contained in even the non-DP chains. We need to account for these nodes belonging to non-DP chain nodes. This is the second step to finding the coverage rate.

Proposition 17 *The coverage rate of a single DP table is expected to be*

$$\mathtt{D}_{cr} = \frac{2}{e^{\hat{t}/t} - 1} \int_0^{\hat{t}/t} \frac{\exp(\Xi u) - 1}{(\Xi+1)\exp(\Xi u) + (\Xi - 1)} \, \exp(u)\, du$$

where $\Xi = \sqrt{1 + \frac{2\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}}}$.

Proof To count the number of distinct non-DPs belonging to all DP chains, we need to subtract the number of all new points belonging to non-DP chains from $\sum_{i=0}^{\hat{t}-1} m_i$. Before doing this, we first need to consider whether these points may not also appear within the DP chains as new points. It is clear that any new node belonging to a non-DP chain cannot have appeared in a column previous to its position, as the node is assumed to be new. Furthermore, such a node cannot appear within the DP chains in the same column or any future columns, since it would then reach a DP before the chain length bound is exceeded. Hence new nodes belonging to non-DP chains do not appear within any DP chains, and we may safely remove all these new points without worrying about their possible contribution to DP chain coverage.

Now, let us count how many points belong to non-DP chains, each column at a time. We start with the 0-th column. Among all m_0 chains, even though we do not know ahead of time which ones they would turn out to be, there will be $m_0(1-\frac{1}{t})^{\hat{t}}$ chains that do not reach a DP even after $\hat{t}$ more iterations. Hence $m_0\big(1-\frac{1}{t}\big)^{\hat{t}}$ nodes among the m_0 nodes belonging to the 0-th column need to be removed from the count of new nodes. As for the 1-st column, we had focused on m_1 chains, but $m_1\big(1-\frac{1}{t}\big)^{\hat{t}-1}$ nodes among these will not reach a DP before exceeding chain length bound, and need to be removed. The general term is now clear.

The coverage rate of a single DP table can now be stated as

$$\frac{1}{mt}\sum_{k=0}^{\hat{t}-1} m_k \Big\{1-\Big(1-\frac{1}{t}\Big)^{\hat{t}-k}\Big\}.$$

Using Lemma 16, we can approximate this to

$$\begin{aligned}
&\frac{1}{mt}\sum_{k=0}^{\hat{t}-1} \mathsf{N}\big(\sigma(k+1)-\sigma(k)\big)\Big\{1-\Big(1-\frac{1}{t}\Big)^{\hat{t}-k}\Big\}\\
&\qquad = \frac{\mathsf{N}}{mt}\frac{1}{t}\sigma(\hat{t}) + \frac{\mathsf{N}}{mt}\sum_{k=0}^{\hat{t}-1}\sigma(k)\Big(1-\frac{1}{t}\Big)^{\hat{t}-k}\frac{1}{t}\\
&\qquad \approx \frac{\sigma(\hat{t})}{\mathtt{D}_{msr}} + \frac{t}{\mathtt{D}_{msr}}\exp\Big(-\frac{\hat{t}}{t}\Big)\sum_{k=0}^{\hat{t}-1}\sigma(k)\exp\Big(\frac{k}{t}\Big)\frac{1}{t}.
\end{aligned}$$

Since the coverage rate is of $O(1)$ order and the first term $\frac{\sigma(\hat{t})}{\mathtt{D}_{msr}}$ is of $O\big(\frac{1}{t}\big)$ order, we simply discard the first term, and the summation term can be approximated by the integration

$$\frac{t}{\mathtt{D}_{msr}}\, e^{-\hat{t}/t}\int_0^{\hat{t}/t}\sigma(tu)\exp(u)\,du,$$

when $\frac{1}{t}$ is small. The claimed formula follows after substitution of $\sigma(tu)$, as given by Lemma 16, and some simplifications. □

We state the $\hat{t} \gg t$ case separately for later use.

Proposition 18 *The expected coverage rate of a single DP table is approximately*

$$\mathtt{D}_{cr} = \frac{2}{\sqrt{1+2\mathtt{D}_{msr}}+1},$$

when the chain length bound $\hat{t}$ is sufficiently large.

Proof When the chain length bound $\hat{t}$ is sufficiently large, almost all of the $m_0 \approx m$ chains that are generated will terminate with a DP, and hence the coverage rate may be computed as $\frac{1}{mt}\sum_{i=0}^{\hat{t}-1} m_i$.

Based on Lemma 16, we may write

$$\mathtt{D}_{cr} = \lim_{\hat{t}\to\infty} \frac{\sum_{i=0}^{\hat{t}-1} m_i}{mt} \approx \lim_{\hat{t}\to\infty} \frac{\mathsf{N}\sigma(\hat{t})}{mt} = \lim_{\hat{t}\to\infty} \frac{2}{1-e^{-\hat{t}/t}} \frac{e^{\Xi\hat{t}/t}-1}{(\Xi+1)e^{\Xi\hat{t}/t}+(\Xi-1)}$$

where $\Xi = \sqrt{1+\frac{2\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}}}$. The limit can be simplified to what is claimed. □

5.2 Time memory tradeoff curve

Our next goal is to summarize the ability of the DP tradeoff algorithm to balance storage against online time in a single tradeoff equation.

We now visualize the chains of the DP matrix as having been aligned at the ending points. The online iterations for the processing of a single DP table are counted starting from the 1-st iteration. That is, checking whether the given password hash $\mathbf{y} = F(\mathbf{x})$ is among the DPs in the DP table is referred to as the 1-st iteration.

Our first task is to find the probability for merges to occur between DP chains.

Lemma 19 *Fix a random function $F : \mathcal{N} \to \mathcal{N}$ and suppose that we are given a pre-computed DP chain of length $j \le \hat{t}$, generated with F from a random non-DP starting point. If a second chain is generated with F from a random starting point, the probability for it to become a DP chain of length i and merge with the given pre-computed chain is*

$$\frac{t}{\mathsf{N}}\Big\{\exp\Big(\frac{\min\{i,j\}}{t}\Big)-1\Big\}\exp\Big(-\frac{i}{t}\Big).$$

Proof Within this proof, let us refer to the event of the second chain becoming a DP chain of length i and merging with the pre-computed chain simply as *the event*.

We first restrict ourselves to the $i \le j$ case and fix notation for the two chains as follows.

$$\begin{array}{llllllll}
\mathbf{x}_0 \to \cdots \to \mathbf{x}_{j-i} & \to \mathbf{x}_{j-i+1} & \to \mathbf{x}_{j-i+2} & \to \cdots \to \mathbf{x}_{j-1} & \to \mathbf{x}_j \\
\qquad\qquad\quad \mathbf{y}_0 & \to \mathbf{y}_1 & \to \mathbf{y}_2 & \to \cdots \to \mathbf{y}_{i-1} & \to \mathbf{y}_i
\end{array}$$

The nodes $\mathbf{x}_0, \dots, \mathbf{x}_{j-1}$ are non-DPs and $\mathbf{x}_j$ is a DP.

Let us consider all possible scenarios by which the event can occur. If the randomly chosen starting point $\mathbf{y}_0$ happens to be equal to $\mathbf{x}_{j-i}$, then the second chain will follows the first chain and the event surely will occur. On the other hand, if either $\mathbf{y}_0$ is one of the points $\mathbf{x}_0, \dots, \mathbf{x}_{j-i-1}, \mathbf{x}_{j-i+1}, \dots, \mathbf{x}_{j-1}$, or is a DP, then the event cannot occur. In the remaining case, i.e. when $\mathbf{y}_0$ is neither a DP nor any one of the points $\mathbf{x}_0, \dots, \mathbf{x}_{j-1}$, then the possibility of the event occurring remains. Furthermore, in this last case, we may freely set $F(\mathbf{y}_0)$ to a randomly chosen point of $\mathcal{N}$.

The above argument may now be repeated. If the randomly chosen $\mathbf{y}_1 = F(\mathbf{y}_0)$ is equal to $\mathbf{x}_{j-i+1}$, then the event occurs. If $\mathbf{y}_1$ is either a DP or one of the points $\mathbf{x}_0, \dots, \mathbf{x}_{j-i}, \mathbf{x}_{j-i+2}, \dots, \mathbf{x}_{j-1}$, then the event cannot occur. And if $\mathbf{y}_1$ is neither a DP nor one of the points $\mathbf{x}_0, \dots, \mathbf{x}_{j-1}$, then the event occurrence is yet undecided and we are free to define $\mathbf{y}_2 = F(\mathbf{y}_1)$ to a random point of $\mathcal{N}$.

Hence, when $i \leq j$, the probability for the event to occur may be written as

$$\frac{1}{\mathsf{N}} + \Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)\frac{1}{\mathsf{N}} + \Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)^2\frac{1}{\mathsf{N}} + \cdots + \Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)^i\frac{1}{\mathsf{N}},$$

which is equal to

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

Noting that $\frac{j}{\mathsf{N}} \ll \frac{1}{t}$ and using $\big(1 - \frac{1}{t}\big)^{i+1} \approx \big(1 - \frac{1}{t}\big)^i \approx \exp\big(-\frac{i}{t}\big)$, we can approximate this as

$$\frac{t}{\mathsf{N}}\Big\{1 - \exp\Big(-\frac{i}{t}\Big)\Big\}.$$

We can similarly work with the $i \geq j$ case. The event can occur only if the beginning random choices $\mathbf{y}_0, \ldots, \mathbf{y}_{i-j-1}$ are made among non-DPs that are different from $\mathbf{x}_0, \ldots, \mathbf{x}_{j-1}$. The probability for the event to occur is

$$\Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)^{i-j}\frac{1}{\mathsf{N}} + \Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)^{i-j+1}\frac{1}{\mathsf{N}} + \cdots + \Big(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\Big)^i\frac{1}{\mathsf{N}},$$

which is approximately

$$\frac{t}{\mathsf{N}}\Big\{\exp\Big(-\frac{i-j}{t}\Big) - \exp\Big(-\frac{i}{t}\Big)\Big\}.$$

The results for the cases $i \leq j$ and $i \geq j$ can now be combined and stated as claimed. $\square$

With the probability of alarms in our hands, we can compute the cost induced by false alarms.

Lemma 20 *The number of extra one-way function invocations induced by alarms is expected to be*

$$t\,\frac{\mathsf{D}_{msr}}{1 - e^{-\hat{t}/t}}\Big\{2 - 8\,e^{-\hat{t}/2t} + \big(5 + 3(\hat{t}/t) - \frac{1}{2}(\hat{t}/t)^2\big)\,e^{-\hat{t}/t} + e^{-2\hat{t}/t}\Big\},$$

for each DP table.

Proof When the chains are generated from $m_0 = \frac{m}{1-e^{-\hat{t}/t}}$ non-DP starting points, one can expect to collect

$$\frac{m}{1 - e^{-\hat{t}/t}}\Big(1 - \frac{1}{t}\Big)^{j-1}\frac{1}{t} \approx \frac{\frac{m}{t}}{1 - e^{-\hat{t}/t}}\,\exp\Big(-\frac{j}{t}\Big) \tag{10}$$

DP chains of length j. The probability of collision between the online chain and any one of these DP chains of length j, at the i-th iteration of the online phase, is given by Lemma 19. Here, the 1-st iteration deals with an online chain of length one, rather than zero, that starts at the unknown password and ends at the password hash. The third component is the work required at each collision. If we take advantage of the fact that there is a chain length bound, in most cases, the number of iterations required to deal with a collision between a pre-computed chain of length j and an online chain of length i will be $\min\{\hat{t} - i + 1, j\}$. The only exception is when a pre-image to the password hash is found, which is rare enough to be ignored.

Multiplying the three components and summing over all possible indices i and j, the expected number of iteration can be expressed as

$$\sum_{i=1}^{\hat{t}} \sum_{j=1}^{\hat{t}} \frac{\frac{m}{t}}{1-e^{-\hat{t}/t}} \exp\Big(-\frac{j}{t}\Big) \cdot \frac{t}{\mathsf{N}} \Big\{ \exp\Big(\frac{\min\{i,j\}}{t}\Big) - 1 \Big\} \exp\Big(-\frac{i}{t}\Big) \cdot \min\{\hat{t}-i+1, j\}.$$

Replacing $\frac{i}{t}$ with u and $\frac{j}{t}$ with v, the above can be approximated by the integration

$$\frac{\frac{mt^2}{\mathsf{N}}\, t}{1-e^{-\hat{t}/t}} \int_0^{\hat{t}/t} \int_0^{\hat{t}/t} \exp(-u)\exp(-v) \Big\{ \exp\big(\min\{u,v\}\big) - 1 \Big\} \min\Big\{ \frac{\hat{t}}{t} - u, v \Big\} \, dv\, du,$$

when $\frac{1}{t}$ is small. The claimed value appears when this definite integral is computed. □

Finally, we write the tradeoff curve for the DP tradeoff in a way that takes the extra cost of false alarms into account.

Theorem 21 *The time memory tradeoff curve for the DP tradeoff is* $TM^2 = \mathtt{D}_{tc}\mathsf{N}^2$*, where the tradeoff coefficient is*

$$\mathtt{D}_{tc} = \Big\{ (2\mathtt{D}_{msr}+1) - \frac{8\mathtt{D}_{msr}}{e^{\hat{t}/2t}} + \frac{(5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2})\mathtt{D}_{msr} - 2}{e^{\hat{t}/t}} + \frac{\mathtt{D}_{msr}+1}{e^{2\hat{t}/t}} \Big\} \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{1-e^{-\hat{t}/t}} \frac{\mathtt{D}_{pc}^2}{\mathtt{D}_{cr}\mathtt{D}_{msr}}.$$

Proof The i-th DP table is processed if and only if all previous tables do not return the correct password. The probability of such a failure is $\big(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\big)^{i-1}$. The time required in processing a single table is the sum of hash invocation counts given by Lemma 11 and Lemma 20. Hence the expected total running time of the DP tradeoff may be written as

$$T = \sum_{i=1}^{l} \Big(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\Big)^{i-1} \Big\{ \big(1-e^{-\hat{t}/t}\big) + \frac{\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}} \Big(2 - \frac{8}{e^{\hat{t}/2t}} + \frac{5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2}}{e^{\hat{t}/t}} + \frac{1}{e^{2\hat{t}/t}}\Big) \Big\}\, t.$$

The summation index i appears only in the first multiplicative factor, and we can easily check that

$$\begin{aligned}\sum_{i=1}^{l} \Big(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\Big)^{i-1} &= \frac{\mathsf{N}}{\mathtt{D}_{cr}mt}\Big\{1-\Big(1-\frac{\mathtt{D}_{cr}mt}{\mathsf{N}}\Big)^{l}\Big\} \\ &\approx \frac{t}{\mathtt{D}_{cr}\mathtt{D}_{msr}}\Big\{1-\exp\Big(-\mathtt{D}_{cr}\frac{mtl}{\mathsf{N}}\Big)\Big\} = \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{\mathtt{D}_{cr}\mathtt{D}_{msr}}\, t.\end{aligned}$$

The running time can now be rewritten as

$$T = \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{\mathtt{D}_{cr}\mathtt{D}_{msr}} \Big\{ \big(1-e^{-\hat{t}/t}\big) + \frac{\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}} \Big(2 - \frac{8}{e^{\hat{t}/2t}} + \frac{5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2}}{e^{\hat{t}/t}} + \frac{1}{e^{2\hat{t}/t}}\Big) \Big\}\, t^2.$$

Since the storage is $M = ml$, we have

$$\begin{aligned}TM^2 &= \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{\mathtt{D}_{cr}\mathtt{D}_{msr}} \Big\{ \big(1-e^{-\hat{t}/t}\big) + \frac{\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}} \Big(2 - \frac{8}{e^{\hat{t}/2t}} + \frac{5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2}}{e^{\hat{t}/t}} + \frac{1}{e^{2\hat{t}/t}}\Big) \Big\}\, (mtl)^2 \\ &= \Big\{ \big(1-e^{-\hat{t}/t}\big) + \frac{\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}} \Big(2 - \frac{8}{e^{\hat{t}/2t}} + \frac{5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2}}{e^{\hat{t}/t}} + \frac{1}{e^{2\hat{t}/t}}\Big) \Big\} \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{\mathtt{D}_{cr}\mathtt{D}_{msr}}\, \mathtt{D}_{pc}^2\mathsf{N}^2,\end{aligned}$$

which is what is claimed. □

The following statement is an immediate consequence of the above theorem.

Corollary 22 *The time memory tradeoff curve for the DP tradeoff is* $M^2T = \mathtt{D}_{tc}\mathsf{N}^2$ *with*

$$\mathtt{D}_{tc} = (2\mathtt{D}_{msr}+1)\big(1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}\big)\,\frac{\mathtt{D}_{pc}^2}{\mathtt{D}_{cr}\mathtt{D}_{msr}},$$

when the chain length bound $\hat{t}$ *is sufficiently large.*

5.3 Efficient use of storage

The storage size M appearing in the tradeoff curves is the total number of starting point and ending point pairs that need to be stored in the tradeoff tables. In practice, it is important to know the actual storage size, or the number of bits, required. Each starting point and ending point pair can surely be stored in $2\log\mathsf{N}$ bits, but there are techniques that achieve more efficient use of storage.

Below, we shall assume a suitable method of enumerating the elements of $\mathscr{N}$ has been fixed and treat elements of $\mathscr{N}$ as $\log\mathsf{N}$-bit integers. This enumeration is trivial when $\mathscr{N}$ is the set of all bit strings of certain length, but may require a small amount of work when $\mathscr{N}$ is given as the set of character passwords with certain complicated structures.

The most widely known technique for storage reduction, other than the trivial fact that the distinguished part of the ending point need not be recorded in the DP table, concerns the choice of starting points. One fixes the starting points[4] to the integers 0 through $m-1$, instead of using randomly chosen points. Then the storage of each starting point will require $\log m$ bits, rather than $\log\mathsf{N}$ bits. If one wishes to use different starting points for each separate table, one can concatenate the table number to the above mentioned integers before using them as starting points. Note that any part that is common within each table need not be stored separately for each starting point. In view of what random functions are, employing such a deterministic way of choosing starting points will not cause any change in the behavior of the tradeoff.

The second method of reducing storage is called the hash table technique. To facilitate fast table lookups, the tradeoff tables are usually sorted on the ending point before being written to storage. Instead of performing the sorting operation, one fixes a hash function[5] that maps elements of $\mathscr{N}$ to $\log m$-bit strings and records each starting point and ending point pair at the position in the table addressed by the hash value of the ending point. There are various ways to deal with collision of addresses. Note that, since the address contains m bits of information, roughly m bits of the ending point can be removed with minimal loss of information. One advantage of this method, in addition to reducing storage, is that, whereas a lookup to a sorted table requires time that is logarithmic in the table size, the hash table technique allows constant time table lookups.

Let us also explain a simplified form of the hash table technique, commonly referred to as the index file method. After sorting the table on the ending points, one focuses on the most significant $\{(\log m)-\varepsilon\}$ bits of each ending point, where ε is a small positive integer. For each integer $0 \le i < \frac{m}{2^\varepsilon}$, we can expect to find 2^ε consecutive entries in the sorted table that have the first $\{(\log m)-\varepsilon\}$ bits of the ending point equal to integer i. One can now remove

[4] To be more precise, this should be 0 through m_0-1, but we will be content with the approximate value.

[5] This is not necessarily a cryptographic hash function, although the same term is used. At the extreme, even truncation to $\log m$ bits may be used.

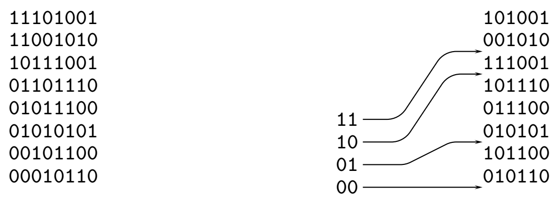


Fig. 1 Index file technique (The sorted list on the LHS is transformed to the RHS list, which contains two less bits per entry.)

$\{(\log m) - \varepsilon\}$ bits of each ending point and provide an index file that points to the starting positions for each i value without loosing any information. The number of entries contained in the index file is only $\frac{m}{2^{\varepsilon}}$ and hence the additional storage required by its introduction can be ignored. An example is illustrated in Figure 1. In practice, the index file often contains the number of entries corresponding to each index value rather than the full physical addresses.

The two methods considered so far reduce storage requirement without any loss of information concerning each starting point and ending point pair. This is not so with the third and final method we explain. In this method one simply truncates a part of the ending point before storage. After each iteration of the online phase, the current end of the online chain is truncated and compared with the truncated ending points of the table. The table lookups may now return a match even when a merge between the online chain and a pre-computation chain did not happen. Still, since we were already expecting false alarms, no new measure needs to be devised to deal with such a new type of false alarms. Aggressive ending point truncation will cause more frequent false alarms, hence it should be balanced with its effect in reducing storage.

The word truncation may give the impression that such a method is applicable only when the space $\mathcal{N}$ consists of bit strings. On spaces that look different, any surjective map that is pre-image uniform, in the sense that the number of pre-images for each element in the range is identical, can serve as the truncation operation. In practice, password hashes are usually bit strings and one does not apply the reduction function at the end of any chain. In fact, DPs are usually defined using the password hashes, rather than the passwords produced through the reduction function.

The ending point truncation method seems to be known among many cryptographers, but no guideline as to how much of the ending point can be truncated can be found. Let us analyze the effects of ending point truncation on the running time of the online phase. Below, we assume that the ending points are truncated in such a way that two random points of $\mathcal{N}$, when truncated in the specified manner, will have probability $\frac{1}{r}$ of agreeing with each other. We shall express such a situation as having $\frac{1}{r}$ probability of truncated match. For example, if $\log t$ bits from the ending points were truncated with $\mathtt{D}_{msr} = 1$, so that $(\log m + \log t)$ bits remain, then the truncated matches would happen with probability $\frac{1}{mt}$. When truncating ending DPs, one should truncate the random-looking part, rather than the distinguished part. Removal of the distinguished part can always be undone, and does not cause any loss of ending point information.

Lemma 23 *Let us assume the use of ending point truncation with the truncated match probability set to $\frac{1}{r}$. The number of extra one-way function invocations induced by false alarms related to ending point truncation is expected to be*

$$t\,\frac{1-2(\hat{t}/t)\,e^{-\hat{t}/t}-e^{-2\hat{t}/t}}{1-e^{-\hat{t}/t}}\,\frac{mt}{r},$$

for each DP table.

Proof Consider a random function $F : \mathcal{N} \to \mathcal{N}$ and suppose that the first chain, generated with F and a random non-DP starting point, produced a DP chain of length $j \le \hat{t}$. Now, suppose a second chain is generated with F from a random non-DP starting point. Let us compute the probability for the second chain to become a DP chain of length i and not merge with the first chain, but have the same truncated ending point as the first chain.

The first i nodes of the second chain must be chosen among non-DPs that are different from the j pre-ending points of the first chain. The i-th node chosen, when truncated, needs to agree with the truncated ending point of the first chain. Note that this agreement already requires the final point to be a DP. Thus the probability we aimed to write can be expressed as

$$\left(1 - \frac{1}{t} - \frac{j}{\mathsf{N}}\right)^i \frac{1}{r} \approx \exp\left(-\frac{i}{t}\right)\frac{1}{r}. \tag{11}$$

Now, we can combine the number of DP chains of length j, as given by (10), together with the probability of non-merging truncated collision with such a chain, as given by (11), to write the cost of truncation related false alarms as

$$\sum_{i=1}^{\hat{t}} \sum_{j=1}^{\hat{t}} \frac{\frac{m}{t}}{1 - e^{-\hat{t}/t}} \ \exp\left(-\frac{j}{t}\right) \cdot \exp\left(-\frac{i}{t}\right) \frac{1}{r} \cdot \min\{\hat{t} - i + 1, j\}.$$

It now suffices to simplify this expression. Replacing $\frac{i}{t}$ with u and $\frac{j}{t}$ with v, the above can be approximated by the definite integral

$$\frac{mt^2}{1 - e^{-\hat{t}/t}} \frac{1}{r} \int_0^{\hat{t}/t} \int_0^{\hat{t}/t} \exp(-u) \exp(-v) \min\left\{\frac{\hat{t}}{t} - u, v\right\} dv\, du,$$

when $\frac{1}{t}$ is small. We arrive at the claimed value when this is explicitly computed. □

Combining Lemma 11, Lemma 20, and Lemma 23, we know that the online processing of a single DP table requires

$$\begin{aligned} & t\left(1 - e^{-\hat{t}/t}\right) && (12)\\ & + t\ \frac{\mathtt{D}_{msr}}{1 - e^{-\hat{t}/t}} \left\{2 - 8 e^{-\hat{t}/2t} + \left(5 + 3(\hat{t}/t) - \frac{1}{2}(\hat{t}/t)^2\right) e^{-\hat{t}/t} + e^{-2\hat{t}/t}\right\} && (13)\\ & + t\ \frac{1}{1 - e^{-\hat{t}/t}} \left\{1 - 2(\hat{t}/t)\, e^{-\hat{t}/t} - e^{-2\hat{t}/t}\right\} \frac{mt}{r}, && (14) \end{aligned}$$

invocations of the one-way function. The first two terms (12) and (13) are both of order $\Theta(t)$. The total cost will remain of the same order if the third term (14), addressing the cost of false alarms related to ending point truncation, is of the same order. One can see that this is equivalent to requiring $\frac{mt}{r} = \Theta(1)$. Now, observe that when $\frac{mt}{r} = \Theta(1)$, increasing the truncated matching probability $\frac{1}{r}$ by a factor of two will have a significant undesirable effect on the total online running time, while the reduction in storage achieved by such an increase is only by a single bit per table entry. Hence we would like the term (14) to become a very small fraction of the sum of the terms (12) and (13). In other words, for each set of parameters t, m, and $\hat{t}$, it is advisable to choose r as small as possible, i.e., truncate as much as possible, under the condition that

$$\frac{mt}{r} \ll \frac{\left(1 - e^{-\hat{t}/t}\right)^2 + \mathtt{D}_{msr}\left\{2 - 8 e^{-\hat{t}/2t} + \left(5 + 3(\hat{t}/t) - \frac{1}{2}(\hat{t}/t)^2\right) e^{-\hat{t}/t} + e^{-2\hat{t}/t}\right\}}{1 - 2(\hat{t}/t)\, e^{-\hat{t}/t} - e^{-2\hat{t}/t}}.$$

All our further discussion will assume such an approach has been taken so that Theorem 21, in particular, remains valid.

Let us summarize the discussion of this subsection. We shall be very rough and ignore all constants of order $\Theta(1)$. Sequential use of starting points allows each starting point to be recorded in approximately $\log m$ bits. One can truncate close to $\log t$ bits from each ending point with minimal effect on the online running time. Of the remaining $\log m + \log t$ bits of the ending point, we do not need to store the $\log t$ bits that is fixed through the distinguishing property. Furthermore, the hash table or index file technique allows us to remove almost $\log m$ more bits without any loss of information. In all, $\log m$ bits are required to store each starting point and a very small number of bits are required to store each ending point.

6 Hellman Tradeoff

In this section, we summarize facts about the Hellman tradeoff. Even though none of these have appeared in the literature in the current form, each fact, other than that concerning the optimal use of storage, is either very easy to prove or is a straightforward consequence of the recent works [9] and [10].

Our first two facts are rather direct consequences of the definitions. The pre-computation effort of the Hellman tradeoff can be expressed as follows.

Proposition 24 *The pre-computation phase of the Hellman tradeoff requires* $\mathtt{H}_{pc}\mathsf{N}$ *one-way function invocations, where the pre-computation coefficient is*

$$\mathtt{H}_{pc} = \mathtt{H}_{msr}\mathtt{H}_{nt}.$$

Proof Since a single Hellman table consists of m chains of length t, its creation requires mt one-way function invocations. The total pre-computation cost is $mtl = mt^2\frac{l}{t} = \mathtt{H}_{msr}\mathtt{H}_{nt}\mathsf{N}$. □

The proof for the next statement is almost identical to that for Proposition 13, and we shall be very brief.

Proposition 25 *The success probability of the Hellman tradeoff is*

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

Proof The success probability expected after completely processing all l tables is

$$1 - \left(1 - \frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^l \approx 1 - \exp\left(-\mathtt{H}_{cr}\frac{mtl}{\mathsf{N}}\right) = 1 - e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}.$$

This argument ignores the fact that the multiple tables are not independent from each other for each randomly chosen function F. □

We next show how to compute the coverage rate, so that the above expression for probability of success can be put to use.

Proposition 26 *The coverage rate of a single Hellman table is expected to be*

$$\mathtt{H}_{cr} = \frac{\sqrt{2}}{\sqrt{\mathtt{H}_{msr}}}\,\frac{e^{\sqrt{2\mathtt{H}_{msr}}} - 1}{e^{\sqrt{2\mathtt{H}_{msr}}} + 1}.$$

Proof The coverage rate of a single Hellman table is given in [10] as

$$\mathtt{H}_{cr} = \frac{2\mathsf{N}}{mt}\left(1-\frac{m}{2\mathsf{N}}\right)\frac{e^{\sqrt{mt^2/2\mathsf{N}}}-e^{-\sqrt{mt^2/2\mathsf{N}}}}{\left(\sqrt{2\mathsf{N}/m}+1\right)e^{\sqrt{mt^2/2\mathsf{N}}}+\left(\sqrt{2\mathsf{N}/m}-1\right)e^{-\sqrt{mt^2/2\mathsf{N}}}}.$$

Since $\frac{m}{2\mathsf{N}} \ll 1$, we can approximate this by ignoring the second multiplicative factor. When the matrix stopping rule $mt^2 = \mathtt{H}_{msr}\mathsf{N}$ is applied, the approximation becomes

$$\mathtt{H}_{cr} = \frac{2t}{\mathtt{H}_{msr}}\,\frac{e^{\sqrt{2\mathtt{H}_{msr}}}-1}{t\sqrt{2/\mathtt{H}_{msr}}\left(e^{\sqrt{2\mathtt{H}_{msr}}}+1\right)+\left(e^{\sqrt{2\mathtt{H}_{msr}}}-1\right)},$$

after some rearrangements. Noting that $\left(e^{\sqrt{2\mathtt{H}_{msr}}}-1\right) \ll t\sqrt{2/\mathtt{H}_{msr}}\left(e^{\sqrt{2\mathtt{H}_{msr}}}+1\right)$, we approximate this once more to

$$\mathtt{H}_{cr} = \frac{2t}{\mathtt{H}_{msr}}\,\frac{e^{\sqrt{2\mathtt{H}_{msr}}}-1}{t\sqrt{2/\mathtt{H}_{msr}}\left(e^{\sqrt{2\mathtt{H}_{msr}}}+1\right)},$$

which is equal to the claimed coverage rate. □

The performance of the Hellman tradeoff is compactly expressed by the following time memory tradeoff curve.

Theorem 27 *The time memory tradeoff curve for the Hellman tradeoff is* $M^2T = \mathtt{H}_{tc}\mathsf{N}^2$, *where the tradeoff coefficient is*

$$\mathtt{H}_{tc} = \left(1+\frac{\mathtt{H}_{msr}}{6}\right)\left(1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}\right)\frac{\mathtt{H}_{pc}^2}{\mathtt{H}_{cr}\mathtt{H}_{msr}}.$$

Proof The i-th Hellman table is processed if and only if all previous tables have failed in returning the correct password. The probability of such a failure is $\left(1-\frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^{i-1}$. The number of one-way function invocations required in processing a single Hellman table may be found in [9], and is $\left(1+\frac{\mathtt{H}_{msr}}{6}\right)t$, with the cost of resolving alarms taken into account. Hence the expected total running time of the Hellman tradeoff may be written as

$$T = \sum_{i=1}^{l}\left(1-\frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^{i-1}\left(1+\frac{\mathtt{H}_{msr}}{6}\right)t.$$

The summation index i appears only in the first multiplicative factor, and we can easily check that

$$\begin{aligned}\sum_{i=1}^{l}\left(1-\frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^{i-1} &= \frac{\mathsf{N}}{\mathtt{H}_{cr}mt}\left\{1-\left(1-\frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^{l}\right\}\\ &\approx \frac{t}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\left\{1-\exp\left(-\mathtt{H}_{cr}\frac{mtl}{\mathsf{N}}\right)\right\} = \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\,t.\end{aligned}$$

The running time can now be rewritten as

$$T = \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\left(1+\frac{\mathtt{H}_{msr}}{6}\right)t^2. \tag{15}$$

Since the storage is $M = ml$, we have

$$M^2T = \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\left(1+\frac{\mathtt{H}_{msr}}{6}\right)(mtl)^2 = \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\left(1+\frac{\mathtt{H}_{msr}}{6}\right)\mathtt{H}_{pc}^2\mathsf{N}^2,$$

which is equal to what is claimed. □

Before continuing, we note that the time T, computed in the proof as (15), counts the number of one-way function computations, and includes the efforts for resolving false alarms. Since the number of table lookups will be smaller, we make this more explicit.

Lemma 28 *The full online processing of the Hellman tradeoff, that use the parameters m, t, and l, is expected to require*

$$t^2 \, \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}$$

lookups to the Hellman tables.

Proof The i-th Hellman table is processed if and only if all previous tables have failed in returning the correct password and processing of each table requires t table lookups. Hence, the expected total number of table lookups is

$$\sum_{i=1}^{l}\left(1-\frac{\mathtt{H}_{cr}mt}{\mathsf{N}}\right)^{i-1} t = \frac{1-e^{-\mathtt{H}_{cr}\mathtt{H}_{pc}}}{\mathtt{H}_{cr}\mathtt{H}_{msr}}\, t^2,$$

as claimed. □

We have so far secured access to the pre-computation cost, the success probability, and the tradeoff performance of the Hellman tradeoff. It remains to discuss the efficient use of storage. The three approaches to storage reduction, discussed in Section 5.3, remain valid for the Hellman tradeoff and an analysis of the ending point truncation method is given below. The concept of probability of truncated match, explained for the DP tradeoff is carried over to the Hellman tradeoff case.

Lemma 29 *Let us assume the use of ending point truncation with the truncated match probability set to $\frac{1}{r}$. The number of extra one-way function invocations induced by truncation related alarms is expected to be*

$$t\,\frac{mt}{2r},$$

for each Hellman table.

Proof Fix a random function $F : \mathscr{N} \to \mathscr{N}$ and suppose that we are given a pre-computed chain of length t, generated with F from a random starting point. Now consider a second chain generated with F from a random starting point. The probability for it to produce an alarm related to truncation, i.e., a truncated ending point match without a merge with the first chain, on the i-th iteration, is

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

This is because the first i nodes of the second chain must be chosen among nodes that are different from the t pre-ending points of the first chain.

Taking account of all m pre-computed chains, the cost induced by the truncation related alarms can now be written as

$$\sum_{i=1}^{t}\frac{m}{r}\cdot(t-i+1) \approx \frac{mt^2}{r}\sum_{i=1}^{t}\left(1-\frac{i}{t}\right)\frac{1}{t}.$$

When $\frac{1}{t}$ is small, by replacing $\frac{i}{t}$ with u, the above can be approximated with the definite integral

$$\frac{mt^2}{r}\int_0^1 (1-u)\,du,$$

which computes to $\frac{mt^2}{2r}$, as claimed. □

Combining this with what we saw during the proof of Theorem 27, the total online time required to deal with a single Hellman table can be stated as

$$t + t\,\frac{\mathtt{H}_{msr}}{6} + t\,\frac{mt}{2r}.$$

The first two terms are of order $\Theta(t)$. After reviewing the arguments concerning ending point truncation for the DP tradeoff, we see that it is advisable to use ending point truncation that truncates as much as possible, while satisfying the condition

$$\frac{mt}{2r} \ll 1 + \frac{\mathtt{H}_{msr}}{6}.$$

Let us summarize the number of bits required to store each starting point and ending point pair. We shall ignore all constants of $\Theta(1)$ order and be very rough. Each starting point requires $\log m$ bits. Ending points may be truncated so that slightly more than $\log m + \log t$ bits remain without visible side-effects on the online running time. The hash table method allows almost $\log m$ additional bits to be saved per ending point without any loss of information. In all, $\log m$ bits are required for each starting point and slightly more than $\log t$ bits are required for each ending point.

7 Rainbow Tradeoff

In this section, we summarize facts about the rainbow tradeoff. The contents appearing in the first half of this section are either very easy to prove or trivial extensions to ideas that have appeared before.

There are two ways of ordering the online phase processing of multiple rainbow tables. In the sequential approach, one fully processes one table before moving onto the next table. In the simultaneous approach, one searches through the same k-th column of all rainbow matrices before moving onto the next column. The simultaneous approach is more efficient in terms of the expected number of one-way function invocations. In practice, handling multiple tables simultaneously may increase the average table lookup time.

In this work, we assume that the l rainbow tables are processed with the simultaneous approach. The 1-st iteration refers to the searching of the password hash $\mathbf{y} = F(\mathbf{x})$ among the ending points of all l tables. The k-th iteration will require $(k-1)\cdot l$ invocations of the one-way function and l lookups to different tables. Columns of the rainbow matrices are numbered from the 0-th, containing the starting points, to the t-th, containing the ending points.

Our first claim is an easy consequence of the relation $mt = \mathtt{R}_{msr}\mathsf{N}$ that defines the notation $\mathtt{R}_{msr}$.

Proposition 30 *The pre-computation phase of the Rainbow tradeoff requires* $\mathtt{R}_{pc}\mathsf{N}$ *one-way function invocations, where the pre-computation coefficient is*

$$\mathtt{R}_{pc} = \mathtt{R}_{msr}\,l.$$

Contents of the following lemma for $l = 1$ were already used in [9], but let us rewrite it here for easy reference. The first statement of this lemma is a trivial extension of a similar statement appearing in [13]. As the proof of the first statement, given in the appendix of [13], makes no mentioning of random functions, we rewrite the proof here within the framework explained in Section 3 of the current paper. The two proofs are essentially the same at the core.[6]

Lemma 31 *The probability for the first k iterations of the online phase to fail is*

$$\prod_{i=1}^{k}\Big(1-\frac{m_{t-i}}{\mathsf{N}}\Big)^{l},$$

where $m_0 = m$ *and* $\frac{m_{i+1}}{\mathsf{N}} = 1-\exp\big(-\frac{m_i}{\mathsf{N}}\big)$. *This product may be approximated by*

$$\left\{\frac{2\mathsf{N}+m(t-k-1)}{2\mathsf{N}+m(t-1)}\,\frac{2\mathsf{N}+m(t-k-2)}{2\mathsf{N}+m(t-2)}\right\}^{l}.$$

Proof The number of distinct nodes expected in each rainbow matrix column is given by the stated m_i. As fully discussed in Section 3.2, there are logical gaps that lead to this claim, but its use as a good approximation can still be justified. In passing, we remark that the different reduction functions used at each column do not remove the logical gap and they do not even provide independence of random function construction between columns.

We can now suppose that a specific one-way function F has been given, and that the rainbow matrix constructed from F contains m_i distinct nodes in the i-th column, for each $0 \le i \le t$. Let $\mathbf{y} = F(\mathbf{x})$ be the inversion target. The i-th iteration of the online phase succeeds if and only if the hidden password $\mathbf{x}$ is located within the $(t-i)$-th column. Assuming that $\mathbf{x}$ was chosen without reference to the rainbow matrix, the i-th iteration fails with probability $\big(1-\frac{m_{t-i}}{\mathsf{N}}\big)$, and all k iterations will fail with the stated probability. Once again, we have ignored the interdependence between columns.

The second statement is based on the approximation

$$\frac{m_i}{\mathsf{N}} \approx \frac{1}{\mathsf{N}/m+i/2}.$$

This is a very small generalization of Theorem 1 from [1], which treats the $m = \mathsf{N}$ case. The proof there can easily be modified to fit the current statement. After rewriting this as

$$1-\frac{m_{t-i}}{\mathsf{N}} \approx \frac{2\mathsf{N}+m(t-i-2)}{2\mathsf{N}+m(t-i)},$$

the sequential cancelations within the product become visible, and we are left with the claimed approximation. □

We can arrive at the next claim by substituting $k = t$ into the above lemma and appropriately approximating the outcome.

Proposition 32 *The success probability of the rainbow tradeoff is*

$$\mathtt{R}_{ps} = 1-\Big(\frac{2}{2+\mathtt{R}_{msr}}\Big)^{2l}.$$

Performance of the rainbow tradeoff is compactly expressed by the following theorem.

[6] Simplification of the approximation given by Lemma 31, for the special case of $m = \mathsf{N}$ and $l = 1$, results in the relation $\prod_{i=c-1}^{t-1}\big(1-\frac{m_i}{\mathsf{N}}\big) \approx \frac{c(c-1)}{t(t+1)}$. We acknowledge that this relation was used multiple times in [1] and that the authors of the paper are likely to have been aware of the statement given here.

Theorem 33 *The time memory tradeoff curve for the rainbow tradeoff is* $M^2T = \mathtt{R}_{tc}\mathsf{N}^2$, *where the tradeoff coefficient is*

$$\mathtt{R}_{tc} = \frac{l^3}{(2l+1)(2l+2)(2l+3)} \begin{pmatrix} \big\{(2l-1)+(2l+1)\mathtt{R}_{msr}\big\}(2+\mathtt{R}_{msr})^2 \\ -4\big\{(2l-1)+l(2l+3)\mathtt{R}_{msr}\big\}\Big(\dfrac{2}{2+\mathtt{R}_{msr}}\Big)^{2l} \end{pmatrix}.$$

Proof Substituting $k = i-1$ into Lemma 31, we know that the i-th iteration is processed with probability

$$\Big\{\frac{2\mathsf{N}+m(t-i)}{2\mathsf{N}+m(t-1)}\,\frac{2\mathsf{N}+m(t-i-1)}{2\mathsf{N}+m(t-2)}\Big\}^l$$
$$\approx \Big\{\Big(1-\frac{m(i-1)}{2\mathsf{N}+m(t-1)}\Big)\Big(1-\frac{m(i-1)}{2\mathsf{N}+m(t-2)}\Big)\Big\}^l \approx \Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l}.$$

The probability of alarm associated with a single chain in a single rainbow matrix at the i-th iteration may be inferred from [9] to be $\frac{i+1}{\mathsf{N}}$. Hence, the expected total running time of the rainbow tradeoff, with false alarms associated with all m rows taken into account, may be written as

$$T = \sum_{i=1}^{t} l\Big\{(i-1)+(t-i+1)\frac{m(i+1)}{\mathsf{N}}\Big\}\Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l}$$
$$\approx t^2 l \sum_{i=1}^{t}\Big\{\frac{i}{t}+\Big(1-\frac{i}{t}\Big)\mathtt{R}_{msr}\frac{i}{t}\Big\}\Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l}\frac{1}{t}.$$

This may be approximated by the definite integral

$$T = t^2 l\int_0^1 u\big\{1+\mathtt{R}_{msr}(1-u)\big\}\Big(1-\frac{\mathtt{R}_{msr}\,u}{2+\mathtt{R}_{msr}}\Big)^{2l}du,$$

which computes to

$$T = t^2 l\,\frac{\begin{pmatrix} \big\{(2l-1)+(2l+1)\mathtt{R}_{msr}\big\}(2+\mathtt{R}_{msr})^2 \\ -4\big\{(2l-1)+l(2l+3)\mathtt{R}_{msr}\big\}\big(\dfrac{2}{2+\mathtt{R}_{msr}}\big)^{2l} \end{pmatrix}}{(2l+l)(2l+2)(2l+3)\mathtt{R}_{msr}^2} \tag{16}$$

It now suffices to combine this with the storage size $M = ml$ and simplify to arrive at the claim. $\square$

It should be noted that the time T appearing in the above tradeoff curve gives the count of one-way function invocations and ignores table lookups.

Lemma 34 *The full online processing of the rainbow tradeoff is expected to require*

$$t\,l\,\frac{2+\mathtt{R}_{msr}-2\big(\frac{2}{2+\mathtt{R}_{msr}}\big)^{2l}}{(2l+1)\mathtt{R}_{msr}}$$

lookups to the rainbow tables.

Proof At the start of proof to Theorem 33, we saw that the i-th iteration is processed with approximate probability

$$\Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l}.$$

Since each iteration requires l table lookups, it suffices to compute

$$\sum_{i=1}^{t} l\,\Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l} \approx t\,l\int_0^1 \Big(1-\frac{\mathtt{R}_{msr}\,u}{2+\mathtt{R}_{msr}}\Big)^{2l} du,$$

to arrive at the expected number of table lookups. □

We now turn to the issue of efficient use of storage. The three approaches to storage reduction, discussed in Section 5.3, remain valid for rainbow tradeoffs and an analysis of the ending point truncation method is given below. The concept of probability of truncated match, used for the DP and Hellman tradeoffs is also carried over to the rainbow tradeoff case.

Lemma 35 *Let us assume the use of ending point truncation with the truncated match probability set to* $\frac{1}{r}$*. The number of extra one-way function invocations induced by false alarms related to ending point truncation is expected to be*

$$t^2 l\,\frac{m}{r}\,\frac{-4+4l\mathtt{R}_{msr}+(2l+1)\mathtt{R}_{msr}^2+4\big(\frac{2}{2+\mathtt{R}_{msr}}\big)^{2l}}{(2l+1)(2l+2)\mathtt{R}_{msr}^2}.$$

Proof For exactly the same reason as given in the proof of Lemma 29, the probability for a randomly generated second chain to produce a truncation induced alarm without merging with the first chain is

$$\Big(1-\frac{1}{\mathsf{N}}\Big)^{i}\Big(\frac{1}{r}-\frac{1}{\mathsf{N}}\Big) \approx \Big(1-\frac{i}{\mathsf{N}}\Big)\Big(\frac{1}{r}-\frac{1}{\mathsf{N}}\Big) \approx \frac{1}{r}.$$

After recalling the probability for the i-th iteration to be processed, and taking account of all the ml pre-computed chains, the expected online cost can be written as

$$\sum_{i=1}^{t}(t-i+1)\cdot\frac{ml}{r}\cdot\Big(1-\frac{\mathtt{R}_{msr}\,\frac{i}{t}}{2+\mathtt{R}_{msr}}\Big)^{2l}.$$

Replacing $\frac{i}{t}$ with u, the above can be approximated by the definite integral

$$\frac{mt^2 l}{r}\int_0^1 (1-u)\Big(1-\frac{\mathtt{R}_{msr}\,u}{2+\mathtt{R}_{msr}}\Big)^{2l} du,$$

when $\frac{1}{t}$ is small, and the claimed value appears when this is computed. □

After reviewing the arguments concerning ending point truncation for the DP and Hellman tradeoffs, we can compare the value given by this lemma against the previously computed main online time (16) to conclude that it is advisable to use ending point truncation that truncates as much as possible, while satisfying the condition

$$\frac{m}{r} \ll \frac{\big\{(2l-1)+(2l+1)\mathtt{R}_{msr}\big\}(2+\mathtt{R}_{msr})^2-4\big\{(2l-1)+l(2l+3)\mathtt{R}_{msr}\big\}\big(\frac{2}{2+\mathtt{R}_{msr}}\big)^{2l}}{(2l+3)\big\{-4+4l\mathtt{R}_{msr}+(2l+1)\mathtt{R}_{msr}^2+4\big(\frac{2}{2+\mathtt{R}_{msr}}\big)^{2l}\big\}}.$$

Note that the righthand side is of $\Theta(1)$ order so that the total online time remains of $\Theta(t^2)$ order when ending point truncation satisfies $\frac{m}{r} = O(1)$.

Let us summarize the number of bits required to store each starting point and ending point pair. We shall ignore all constants of $\Theta(1)$ order and be very rough. Each starting point requires $\log m$ bits. Ending points may be truncated so that slightly more than $\log m$ bits remain without visible side-effects on the online running time. The hash table method allows most of the remaining $\log m$ bits to be removed from the ending point without any loss of information. In all, $\log m$ bits are required for each starting point and only a very small number of bits are required for each ending point.

8 Optimal Tradeoff Parameters

In this section, we shall find optimal parameter sets for each of the three tradeoff algorithms.

Let us present our initial arguments in terms of the Hellman tradeoff. The balance between time and memory achievable by the Hellman tradeoff is expressed by the curve $M^2T = \mathtt{H}_{tc}\mathsf{N}^2$. It is clear that the Hellman algorithm at parameters that bring about a smaller tradeoff coefficient $\mathtt{H}_{tc}$ will require less resources to run. In other words, tradeoff coefficient $\mathtt{H}_{tc}$ is a measure of the tradeoff efficiency, with a smaller value representing better tradeoff performance.

The tradeoff coefficient $\mathtt{H}_{tc}$ is determined by the parameters m, t, and l. It should first be noticed that a better tradeoff coefficient should always be achievable, if one decides to sacrifice the success probability for finding the correct password. Hence, any comparison between two Hellman tradeoff coefficients, achievable through two different sets of parameters, should be done under the condition that they produce the same success probability.

Arguments similar to the above may be made for the DP and rainbow tradeoffs. Hence, for each of the three algorithms, we shall work to find the smallest tradeoff coefficient achievable under a fixed requirement on the success rate. This is not yet a comparison between different algorithms, but only a study of optimal tradeoff coefficient for each separate algorithm. Such an analysis may seem interesting in view of optimal usage of tradeoff algorithms, but can be of limited value in practice. Parameters achieving better tradeoff performance may require more pre-computation, and with large scale implementations of the tradeoff technique, lowering the pre-computation cost may be significantly more valuable than achieving better tradeoff performance. Our purpose of locating the optimal tradeoff parameters is for its use in the next section, where we compare between different algorithms.

8.1 DP tradeoff

The parameter set that achieves optimal DP tradeoff performance, under a fixed requirement on the probability of success, is given below.

Proposition 36 *Let $0 < \mathtt{D}_{ps} < 1$ be any fixed value. The DP tradeoff, under any set of parameters m, t, l, and $\hat{t}$, that are subject to the relations*

$$mt^2 = 1.26453\,\mathsf{N}, \quad l = 1.28007\,\ln\left(\frac{1}{1-\mathtt{D}_{ps}}\right)t, \quad \textit{and} \quad \hat{t} = 2.59169\,t,$$

attains the given value $\mathtt{D}_{ps}$ as its probability of success, and exhibits tradeoff performance corresponding to

$$\mathtt{D}_{tc} = 5.49370\,\mathtt{D}_{ps}\big\{\ln(1-\mathtt{D}_{ps})\big\}^2,$$

as the four parameters are varied. Under any such choice of parameters, the number of one-way function invocations required for the pre-computation phase is

$$\mathtt{D}_{pc} = 1.61869 \ln\left(\frac{1}{1-\mathtt{D}_{ps}}\right),$$

in multiples of N.

The three relations restricting the parameter choices give optimal parameters in the sense that no choice of m, t, l, *and* $\hat{t}$ *can lead to a tradeoff coefficient smaller than the above while achieving* $\mathtt{D}_{ps}$ *as its probability of success.*

Proof Proposition 12 and Proposition 13 state the probability of success for DP tradeoffs as

$$\mathtt{D}_{ps} = 1 - e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}} = 1 - e^{-\mathtt{D}_{cr}\mathtt{D}_{msr}\mathtt{D}_{nt}}.$$

Recalling the definition of $\mathtt{D}_{nt} = \frac{l}{t}$, this relation may equivalently be stated as

$$l = \frac{1}{\mathtt{D}_{cr}\mathtt{D}_{msr}} \ln\left(\frac{1}{1-\mathtt{D}_{ps}}\right) t. \tag{17}$$

Now, referencing Proposition 17, we know that the DP coverage rate $\mathtt{D}_{cr} = \mathtt{D}_{cr}[\mathtt{D}_{msr}, \hat{t}/t]$ may be treated as a function of the two variables $\mathtt{D}_{msr}$ and $\frac{\hat{t}}{t}$. Hence, given any m, t, $\hat{t}$, and $\mathtt{D}_{ps}$, if we set $\mathtt{D}_{msr} = \frac{mt^2}{\mathsf{N}}$ and $\mathtt{D}_{cr} = \mathtt{D}_{cr}[\mathtt{D}_{msr}, \hat{t}/t]$, and also fix l through relation[7] (17), then the DP tradeoff with these parameters will always achieve success probability of $\mathtt{D}_{ps}$.

Keeping in mind that we may freely choose m, t, and $\hat{t}$, and still obtain any requested success probability, we now work to minimize the DP tradeoff coefficient

$$\mathtt{D}_{tc} = \left\{ (2\mathtt{D}_{msr}+1) - \frac{8\mathtt{D}_{msr}}{e^{\hat{t}/2t}} + \frac{(5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2})\mathtt{D}_{msr}-2}{e^{\hat{t}/t}} + \frac{\mathtt{D}_{msr}+1}{e^{2\hat{t}/t}} \right\} \frac{1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}}{1-e^{-\hat{t}/t}} \frac{\mathtt{D}_{pc}^2}{\mathtt{D}_{cr}\mathtt{D}_{msr}},$$

as given by Theorem 21. After some regrouping of variables, we can rewrite this as

$$\begin{aligned} \mathtt{D}_{tc} &= \mathtt{D}_{tmp}\left[\mathtt{D}_{msr}, \frac{\hat{t}}{t}\right] \cdot \left(1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}\right)\left(-\mathtt{D}_{cr}\mathtt{D}_{pc}\right)^2 \\ &= \mathtt{D}_{tmp}\left[\mathtt{D}_{msr}, \frac{\hat{t}}{t}\right] \cdot \mathtt{D}_{ps}\left\{\ln(1-\mathtt{D}_{ps})\right\}^2, \end{aligned}$$

where

$$\begin{aligned} \mathtt{D}_{tmp}\left[\mathtt{D}_{msr}, \frac{\hat{t}}{t}\right] = &\left\{ (2\mathtt{D}_{msr}+1) - \frac{8\mathtt{D}_{msr}}{e^{\hat{t}/2t}} + \frac{(5+\frac{3\hat{t}}{t}-\frac{\hat{t}^2}{2t^2})\mathtt{D}_{msr}-2}{e^{\hat{t}/t}} + \frac{\mathtt{D}_{msr}+1}{e^{2\hat{t}/t}} \right\} \\ &\times \frac{1}{\left(1-e^{-\hat{t}/t}\right)} \frac{1}{\mathtt{D}_{cr}^3\left[\mathtt{D}_{msr}, \frac{\hat{t}}{t}\right]} \frac{1}{\mathtt{D}_{msr}}. \end{aligned} \tag{18}$$

It is clear that, when the probability of success requirement is fixed, minimizing $\mathtt{D}_{tc}$ is equivalent to finding the minimum of $\mathtt{D}_{tmp}[\mathtt{D}_{msr}, \hat{t}/t]$. Note that, even though $\mathtt{D}_{msr} = \frac{mt^2}{\mathsf{N}}$ and $\hat{t}/t$ share the parameter t, since we are free to set m, t, and $\hat{t}$ to any value, there are enough degrees of freedom, and we may treat $\mathtt{D}_{msr}$ and $\hat{t}/t$ as independent variables when looking for the minimum of $\mathtt{D}_{tmp}[\mathtt{D}_{msr}, \hat{t}/t]$.

[7] Note that l must be set to an integer. Since the RHS of (17) is rather large, the error to the success probability, introduced by taking the nearest integer to the RHS value, will be very small.

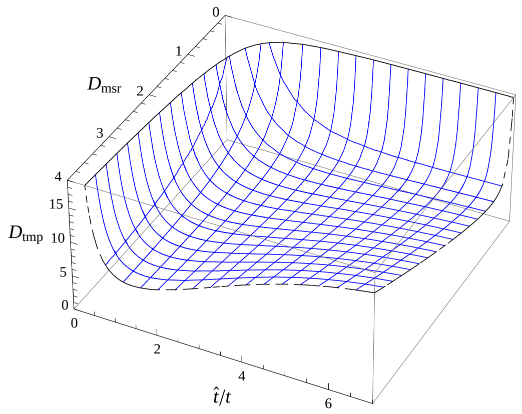


Fig. 2 Tradeoff coefficient for DP tradeoff at fixed probability of success ($\mathtt{D}_{tmp} = \frac{\mathtt{D}_{tc}}{\mathtt{D}_{ps}\cdot\{\ln(1-\mathtt{D}_{ps})\}^2}$)

After substituting $\mathtt{D}_{cr}[\mathtt{D}_{msr}, \hat{t}/t]$, as given by Proposition 17, into the righthand side of (18), we can use numerical methods to find its minimum. One discovers that the minimum value of $\mathtt{D}_{tmp} = 5.49370$ is obtained at $\mathtt{D}_{msr} = 1.25453$ and $\hat{t}/t = 2.59169$. The claimed relation between l and t follows from (17), after evaluation of $\frac{1}{\mathtt{D}_{cr}[\mathtt{D}_{msr},\hat{t}/t]\,\mathtt{D}_{msr}}$ at $\mathtt{D}_{msr} = 1.25453$ and $\hat{t}/t = 2.59169$. The final claim concerning the pre-computation cost follows from an evaluation based on Proposition 12. □

The parameter set achieving minimum tradeoff coefficient for the DP tradeoff is visible through Figure 2. It plots $\mathtt{D}_{tmp} = \frac{\mathtt{D}_{tc}}{\mathtt{D}_{ps}\{\ln(1-\mathtt{D}_{ps})\}^2}$, which is given by (18), as a function of variables $\mathtt{D}_{msr}$ and $\hat{t}/t$.

The tradeoff curve, as given by this proposition, allows us to say more about the tradeoff than the previously known rough curve of $M^2T \approx \mathsf{N}^2$. Suppose that, for some fixed set of parameters, the success rate of the DP tradeoff is not too small, and suppose that one wishes to increase the success rate, to the extent that the failure rate becomes the square of its current value. Then, for optimal choice of parameters, the $\mathtt{D}_{ps}$ factor will change little and the $\{\ln(1-\mathtt{D}_{ps})\}^2$ factor will increase by a factor of four. Hence, one must allow an increase in the online time by a factor of four or use twice the current storage. The proposition also shows that one must endure twice the pre-computation cost to achieve this aim. Of course, the simplest way of doing this would be to increase the number of tables by twice, while keeping all other parameters the same.

While the above result gives the parameters that achieves the optimal tradeoff performance, in practical applications, pre-computation is very costly and one is more likely to choose a sufficiently large $\hat{t}$, so as not to discard any of the pre-computed results.

Proposition 37 *Let $0 < \mathtt{D}_{ps} < 1$ be any fixed value. When the use of $\hat{t} \gg t$ is assumed, the DP tradeoff, under any set of parameters m, t, and l, that are subject to the relations*

$$mt^2 = 0.562047\,\mathsf{N} \quad \textit{and} \quad l = 2.18614\,\ln\Big(\frac{1}{1-\mathtt{D}_{ps}}\Big)\,t,$$

attains the given value $\mathtt{D}_{ps}$ as its probability of success, and exhibits tradeoff performance corresponding to

$$\mathtt{D}_{tc} = 7.01057\,\mathtt{D}_{ps}\big\{\ln(1-\mathtt{D}_{ps})\big\}^2,$$

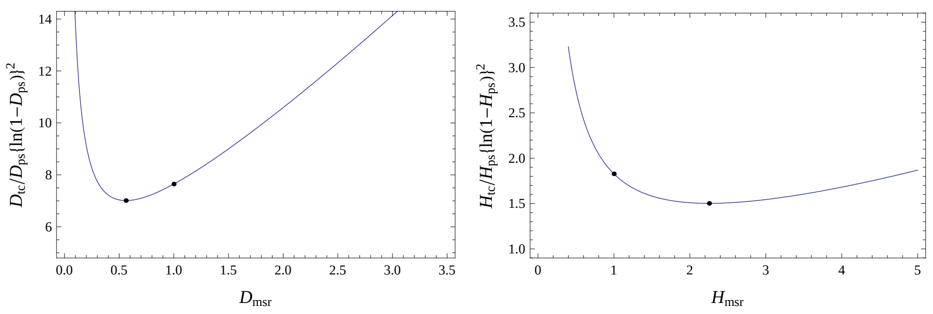


Fig. 3 Tradeoff coefficients for DP tradeoff with $\hat{t} \gg t$ and Hellman tradeoff at fixed probability of success

as the three parameters are varied. Under any such choice of parameters, the number of one-way function invocations required for the pre-computation phase is

$$\mathtt{D}_{pc} = 1.22871 \ln\Big(\frac{1}{1-\mathtt{D}_{ps}}\Big),$$

in multiples of N.

The two relations restricting the parameter choices give optimal parameters in the sense that, as long as $\hat{t} \gg t$ is assumed, no choice of m, t, and l can lead to a tradeoff coefficient smaller than the above while achieving $\mathtt{D}_{ps}$ as its probability of success.

Proof The proof is almost identical to that of Proposition 36. The only difference is that we refer to Proposition 18 to view $\mathtt{D}_{cr}$ as a function of $\mathtt{D}_{msr}$ and obtain the tradeoff coefficient from Corollary 22. Through some regrouping of terms we can write

$$\mathtt{D}_{tc} = \Big(2+\frac{1}{\mathtt{D}_{msr}}\Big)\frac{1}{\mathtt{D}_{cr}^3}\left(1-e^{-\mathtt{D}_{cr}\mathtt{D}_{pc}}\right)\left(\mathtt{D}_{cr}^2\mathtt{D}_{pc}^2\right)$$

and by substituting $\mathtt{D}_{cr}$ into the appropriate part of $\mathtt{D}_{tc}$, we obtain

$$\mathtt{D}_{tc} = \Big(2+\frac{1}{\mathtt{D}_{msr}}\Big)\Big(\frac{\sqrt{1+2\mathtt{D}_{msr}}+1}{2}\Big)^3 \mathtt{D}_{ps}\left\{\ln(1-\mathtt{D}_{ps})\right\}^2. \tag{19}$$

It suffices to minimize

$$\mathtt{D}_{tmp}[\mathtt{D}_{msr}] := \frac{\mathtt{D}_{tc}}{\mathtt{D}_{ps}\{\ln(1-\mathtt{D}_{ps})\}^2} = \Big(2+\frac{1}{\mathtt{D}_{msr}}\Big)\Big(\frac{\sqrt{1+2\mathtt{D}_{msr}}+1}{2}\Big)^3,$$

seen a function of the single variable $\mathtt{D}_{msr}$. □

In comparison to the previous optimal parameters that utilizes $\hat{t}$ as a free variable, this version shows less efficient tradeoff performance, but requires less pre-computation. The behavior of the DP tradeoff coefficient with $\hat{t} \gg t$, under a fixed requirement for success rate is given as the lefthand side graph of Figure 3. The point of minimum tradeoff coefficient is marked, together with the position corresponding to the more common matrix stopping rule of $\mathtt{D}_{msr} = 1$.

8.2 Hellman tradeoff

We now turn to the Hellman tradeoffs. This is very similar to the DP tradeoff case that uses a sufficiently large $\hat{t}$.

Proposition 38 *Let $0 < \mathtt{H}_{ps} < 1$ be any fixed value. The Hellman tradeoff, under any set of parameters m, t, and l, that are subject to the relations*

$$mt^2 = 2.25433\ \mathsf{N} \quad \textit{and} \quad l = 0.598941 \ln\Big(\frac{1}{1-\mathtt{H}_{ps}}\Big)\,t,$$

attains the given $\mathtt{H}_{ps}$ as its probability of success, and exhibits the tradeoff performance corresponding to

$$\mathtt{H}_{tc} = 1.50217\ \mathtt{H}_{ps}\big\{\ln(1-\mathtt{H}_{ps})\big\}^2,$$

as the three parameters are varied. Under any such choice of parameters, the number of one-way function invocations required for the pre-computation phase is

$$\mathtt{H}_{pc} = 1.35021 \ln\Big(\frac{1}{1-\mathtt{H}_{ps}}\Big),$$

in multiples of N.

The two relations restricting the parameter choices give optimal parameters in the sense that no choice of m, t, and l can lead to a tradeoff coefficient smaller than the above while achieving $\mathtt{H}_{ps}$ as its probability of success.

Proof Since the proof is similar to those of Proposition 36 and Proposition 37 we shall be concise. Based on Proposition 24 and Proposition 25, we may claim $l = \frac{1}{\mathtt{H}_{cr}\,\mathtt{H}_{msr}} \ln\big(\frac{1}{1-\mathtt{H}_{ps}}\big)\,t$. Reference to Proposition 26 shows that the Hellman coverage rate $\mathtt{H}_{cr} = \mathtt{H}_{cr}[\mathtt{H}_{msr}]$ may be seen as a function of $\mathtt{H}_{msr} = \frac{mt^2}{\mathsf{N}}$. Hence, given any m, t, and $\mathtt{H}_{ps}$, we can set l to an appropriate value with which the Hellman tradeoff achieves success probability of $\mathtt{H}_{ps}$.

We now work to minimize the Hellman tradeoff coefficient. The content of Theorem 27 may be rewritten as

$$\mathtt{H}_{tc} = \Big(\frac{1}{\mathtt{H}_{msr}} + \frac{1}{6}\Big)\frac{1}{\mathtt{H}_{cr}^3}\,\mathtt{H}_{ps}\big\{\ln(1-\mathtt{H}_{ps})\big\}^2$$

and substitution of $\mathtt{H}_{cr}$ results in

$$\mathtt{H}_{tc} = \Big(\frac{1}{\mathtt{H}_{msr}} + \frac{1}{6}\Big)\Big(\frac{\sqrt{\mathtt{H}_{msr}}}{\sqrt{2}}\ \frac{e^{\sqrt{2\mathtt{H}_{msr}}}+1}{e^{\sqrt{2\mathtt{H}_{msr}}}-1}\Big)^3\,\mathtt{H}_{ps}\big\{\ln(1-\mathtt{H}_{ps})\big\}^2. \tag{20}$$

For a fixed success probability, it suffices to minimize

$$\mathtt{H}_{tmp}[\mathtt{H}_{msr}] := \frac{\mathtt{H}_{tc}}{\mathtt{H}_{ps}\{\ln(1-\mathtt{H}_{ps})\}^2} = \Big(\frac{1}{\mathtt{H}_{msr}} + \frac{1}{6}\Big)\Big(\frac{\sqrt{\mathtt{H}_{msr}}}{\sqrt{2}}\ \frac{e^{\sqrt{2\mathtt{H}_{msr}}}+1}{e^{\sqrt{2\mathtt{H}_{msr}}}-1}\Big)^3, \tag{21}$$

which is a function of the single variable $\mathtt{H}_{msr}$.

One can use numeric methods to identify the minimum value $\mathtt{H}_{tmp} = 1.50217$, which appears at $\mathtt{H}_{msr} = 2.25433$. The two remaining constants appearing in the proposition may now be obtained through appropriate evaluations. □

The original Hellman tradeoff, which is set to use $mt^2 = \mathsf{N}$ and $l = t$ attains a success probability of 57.68% and the tradeoff curve $M^2T = 0.7797\mathsf{N}^2$, when the cost of false alarms are taken into account. In comparison, the choice of $mt^2 = 2.2543\mathsf{N}^2$ and $l = 0.5160t$, recommended by Proposition 38, gives $M^2T = 0.6409\mathsf{N}^2$, while achieving the same success rate. This is visible through the righthand side graph of Figure 3, where the two dots mark the two parameter choices we have discussed.

The price paid for this better tradeoff performance is the moderate increase in pre-computation from N to $1.1630\mathsf{N}$. Indeed, after combining Proposition 25 and Proposition 26 into

$$\mathtt{H}_{pc} = -\frac{\ln(1-\mathtt{H}_{ps})}{\mathtt{H}_{cr}[\mathtt{H}_{msr}]} = \ln\Big(\frac{1}{1-\mathtt{H}_{ps}}\Big)\frac{\sqrt{\mathtt{H}_{msr}}}{\sqrt{2}}\;\frac{e^{\sqrt{2\mathtt{H}_{msr}}}+1}{e^{\sqrt{2\mathtt{H}_{msr}}}-1}, \tag{22}$$

one can check that the pre-computation $\mathtt{H}_{pc}[\mathtt{H}_{msr}]$ required under any fixed probability of success is an increasing function of $\mathtt{H}_{msr}$. Hence, while any point that is situated to the left of the minimal point may not be optimal in view of tradeoff performance, it corresponds to less pre-computation. Depending on the available computational resources, one may choose to lower pre-computation cost rather than increase the tradeoff efficiency. On the other hand, increasing $\mathtt{H}_{msr}$ beyond the minimizing value 2.25433 will have bad effects on both the pre-computation and the tradeoff performance and should be avoided.

Let us briefly return to the DP tradeoff that uses sufficiently large $\hat{t}$. By combining Proposition 13 and Proposition 18, we can write

$$\mathtt{D}_{pc} = -\frac{\ln(1-\mathtt{D}_{ps})}{\mathtt{D}_{cr}[\mathtt{D}_{msr}]} = \ln\Big(\frac{1}{1-\mathtt{D}_{ps}}\Big)\;\frac{\sqrt{1+2\mathtt{D}_{msr}}+1}{2}, \tag{23}$$

and, as with the Hellman tradeoff, confirm that $\mathtt{D}_{pc}$ is an increasing function of $\mathtt{D}_{msr}$. Since we know from Proposition 37 that the best performance is achieved at $\mathtt{D}_{msr} = 0.562047$, the choice of $\mathtt{D}_{msr} < 0.562047$ may be reasonable in view of lowering pre-computation cost, but using $\mathtt{D}_{msr} > 0.562047$ should be avoided. In particular, the use of $\mathtt{D}_{msr} = 1$ cannot be justified.

8.3 Rainbow tradeoff

The analyses of optimal parameters for the DP and Hellman tradeoffs were very similar. Rainbow tradeoff does not allow the same approach because we have less control over the parameter l. The number of tables l used with DP and Hellman tradeoffs are quite large and we had treated l as if it were a continuous variable. In the rainbow tradeoff case, the table count is usually a small integer and we must keep in mind that it takes only discrete values.

Let us start with a fixed number of tables l. For any given requirement on the success rate, we can rewrite Proposition 32 as

$$\mathtt{R}_{msr} = 2\Big\{\Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{2l}} - 1\Big\} \tag{24}$$

and understand this as a lower bound on $\mathtt{R}_{msr}$ that can be used. It is clear that increasing $\mathtt{R}_{msr}$ under a fixed l will increase the pre-computation cost $\mathtt{R}_{msr}l\mathsf{N}$. One can also work with the tradeoff coefficient $\mathtt{R}_{tc}$, as provided by Theorem 33, to confirm that increasing $\mathtt{R}_{msr}$ under a fixed l will reduce the tradeoff efficiency. Hence, under any fixed l, the exact value of $\mathtt{R}_{msr}$, suggested by (24), should be used to achieve the required success rate.

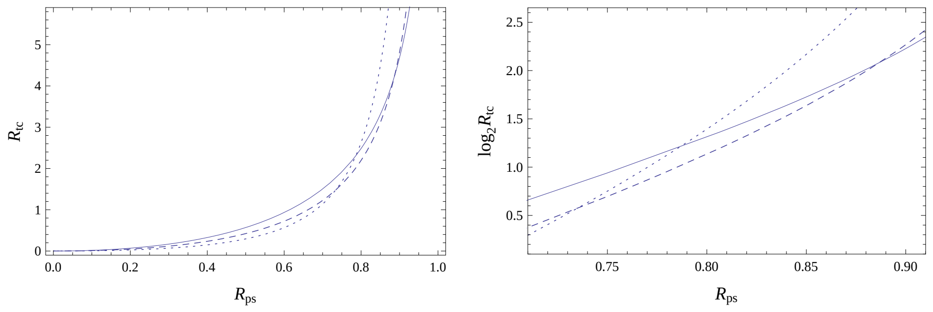


Fig. 4 Tradeoff coefficient of rainbow tradeoff as a function of success rate requirement at small number tables ($l = 1$: dotted, $l = 2$: dashed, $l = 3$: solid)

We can now treat $\mathtt{R}_{msr}$ as a function of the success rate requirement $\mathtt{R}_{ps}$, for any fixed l. After substituting $\mathtt{R}_{msr}$, as given by (24), into the tradeoff coefficient of Theorem 33, one can rewrite it as

$$\begin{aligned}\mathtt{R}_{tc} = \; & \frac{4\,l^3}{(2l+1)(2l+2)(2l+3)} \\ & \times \left(\begin{array}{l} \Big\{ -(2l+3) + 2(2l+1)\Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{2l}} \Big\} \Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{l}} \\ + \Big\{ (2l+1)^2 - 2l(2l+3)\Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{2l}} \Big\} (1-\mathtt{R}_{ps}) \end{array} \right). \end{aligned} \tag{25}$$

For each fixed l, this is a function of the single variable $\mathtt{R}_{ps}$. A plot of this is given as Figure 4 for table counts $l = 1$, 2, and 3. The the righthand side box is a magnified partial view of the lefthand side box in logarithmic scale.

Recalling that a smaller tradeoff coefficient implies better tradeoff performance, one can clearly read from the figure that the use of $l = 1$ is optimal when the requirement for success rate is very low and that the use of successively higher number of tables becomes optimal as the success rate requirement is made more stringent. We have numerically solved for the explicit probabilities at which the transition to the next table count should be made and have recorded this in Table 1.

Let us briefly explain the content of the table with examples. Suppose one aims to achieve the success probability of 99.9% with the rainbow tradeoff. Since 0.999 sits between 0.998775 and 0.999314, it is optimal to use ten tables. If one is requested to set the probability of failure to $\frac{1}{2^7}$, we locate -7 between -6.17353 and -7.08171 and conclude that six tables would be optimal. To understand the other three columns of the table, let us focus on the row that sits between $l = 1$ and $l = 2$. The use of a single table with $\mathtt{R}_{msr} = 1.87905$, or the use two tables at $\mathtt{R}_{msr} = 0.785335$ will both result in the optimal tradeoff coefficient of $\mathtt{R}_{tc} = 1.48026 = 2^{0.565848}$ and success rate 73.4166%.

Note that any given success rate requirement $\mathtt{R}_{ps}$ makes a certain number of tables l as optimal, and the l value fixes $\mathtt{R}_{msr}$ through (24). Since the tradeoff coefficient of Theorem 33 is already determined by l and $\mathtt{R}_{msr}$, and since the relation (24) guarantees $\mathtt{R}_{ps}$ success rate, any parameter set satisfying the mentioned restriction will be optimal in view of the tradeoff coefficient. Let us gather what we have discussed in a proposition.

Table 1 Range of success probability requirements for which each table count l is optimal

l	$\mathtt{R}_{ps}$	$\log_2(1-\mathtt{R}_{ps})$	$\log_2 \mathtt{R}_{tc}$	$\mathtt{R}_{msr}[\mathtt{R}_{ps}, l\uparrow]$	$\mathtt{R}_{msr}[\mathtt{R}_{ps}, l\downarrow]$
	0	0	0		
1					
	0.734166	-1.91140	0.565848	1.87905	0.785335
2					
	0.886651	-3.14116	2.08082	1.44688	0.874929
3					
	0.946562	-4.22600	2.88968	1.25878	0.884357
4					
	0.973305	-5.22729	3.41666	1.14577	0.873341
5					
	0.986146	-6.17353	3.79818	1.06812	0.856920
6					
	0.992618	-7.08171	4.09387	1.01079	0.839893
7					
	0.995992	-7.96295	4.33425	0.966542	0.823891
8					
	0.997795	-8.82486	4.53663	0.931326	0.809415
9					
	0.998775	-9.67274	4.71157	0.902658	0.796529
10					
	0.999314	-10.5104	4.86585	0.878902	0.785129
11					
	0.999614	-11.3404	5.00406	0.858929	0.775059
12					
	0.999782	-12.1649	5.12941	0.841927	0.766150
13					
	0.999877	-12.9850	5.24421	0.827299	0.758246
14					
	0.999930	-13.8020	5.35019	0.814594	0.751208
15					
	0.999960	-14.6163	5.44869	0.803466	0.744914

Proposition 39 *Let $0 < \mathtt{R}_{ps} < 1$ be any given fixed value. Locate the table count l from Table 1 that corresponds to the given $\mathtt{R}_{ps}$ and compute*

$$\mathtt{R}_{msr} = 2\Big\{\Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{2l}} - 1\Big\}.$$

Then the rainbow tradeoff that uses the located l and any parameters m and t satisfying the relation

$$mt = \mathtt{R}_{msr}\mathsf{N}$$

attains the given value $\mathtt{R}_{ps}$ as its probability of success. The tradeoff performance corresponding to

$$\mathtt{R}_{tc} = \frac{l^3}{(2l+1)(2l+2)(2l+3)} \begin{pmatrix} \big\{(2l-1)+(2l+1)\mathtt{R}_{msr}\big\}(2+\mathtt{R}_{msr})^2 \\ -4\big\{(2l-1)+l(2l+3)\mathtt{R}_{msr}\big\}(1-\mathtt{R}_{ps}) \end{pmatrix},$$

can be observed as m and t are varied under the restriction. With any such choice of parameters, the number of one-way function invocations required for the pre-computation phase is

$$\mathtt{R}_{pc} = \mathtt{R}_{msr}\, l,$$

in multiples of N.

The choice of l through Table 1 and the single relation concerning m and t lead to optimal parameters in the sense that no choice of m, t, and l can result in a tradeoff coefficient smaller than the above while achieving $\mathtt{R}_{ps}$ as its probability of success.

To be strictly logical, one must also consider the possibility that allowing the multiple tables to be of different sizes may lead to better tradeoff coefficients. We have analyzed [12] the case of three tables with the most general table sizes and came to the conclusion that optimal tradeoff performance is achieved at equal sized tables. Our method of analyzing this possibility can probably be extended to larger number of tables, but the computations will be much more complicated than what was presented here. Since the examination of the 3-table case showed that we are not likely to gain anything from the more general analysis, we chose to work with equal sized tables. In contrast, for the case of perfect rainbow tables, we have reasons to believe that this extra flexibility will bring about better tradeoff performance.

Finally, we want to provide an argument that is analogous to what was discussed at the end of Section 8.2. One can check that

$$\mathtt{R}_{pc} = \mathtt{R}_{msr}\, l = 2l\Big\{\Big(\frac{1}{1-\mathtt{R}_{ps}}\Big)^{\frac{1}{2l}} - 1\Big\} \tag{26}$$

is a decreasing function of l, for each fixed $\mathtt{R}_{ps}$. Hence, use of an l count that is larger than what is suggested by Table 1 will decrease the pre-computation requirement at the cost of reduced tradeoff efficiency. This may be preferable in some situations. On the other hand, use of an l count that is smaller than the optimal count will have bad effects on both the pre-computation cost and tradeoff efficiency, and should be avoided.

9 Comparison of tradeoff performances

All the tools required for a fair comparison of performances between the tradeoff algorithms are now ready.

9.1 Conversion of the tradeoff coefficients to a common unit

Discussion of the previous section has made it clear that for any comparison of tradeoff algorithms to be fair, the algorithms must be made to present the same probability of success. One must also consider the pre-computation cost required by each algorithm, but this aspect will be considered later. We are also aware that the tradeoff coefficient is a measure of tradeoff performance. Hence let us assume that the DP, Hellman, and rainbow tradeoff algorithms display the respective tradeoff curves

$$M_{\mathtt{D}}^2 T_{\mathtt{D}} = \mathtt{D}_{tc}\mathsf{N}^2, \quad M_{\mathtt{H}}^2 T_{\mathtt{H}} = \mathtt{H}_{tc}\mathsf{N}^2, \quad \text{and} \quad M_{\mathtt{R}}^2 T_{\mathtt{R}} = \mathtt{R}_{tc}\mathsf{N}^2, \tag{27}$$

at the same success rate. We will discuss how to interpret the ratios $\mathtt{D}_{tc} : \mathtt{H}_{tc}$, $\mathtt{D}_{tc} : \mathtt{R}_{tc}$, and $\mathtt{H}_{tc} : \mathtt{R}_{tc}$ of the tradeoff coefficients as ratios of tradeoff performances.

Unit for T. Let us fist consider the time variable T. The appearance of the time value $T_{\mathtt{D}}$ in a DP tradeoff curve signifies that there are parameters $t_{\mathtt{D}}$, $m_{\mathtt{D}}$, $l_{\mathtt{D}}$, and $\hat{t}_{\mathtt{D}}$ with which the DP algorithm will display running time corresponding to $T_{\mathtt{D}}$. To be more exact, the expected online execution time will be that consumed by $T_{\mathtt{D}} = \Theta(t_{\mathtt{D}}^2)$ invocations of the one-way function and at most $l_{\mathtt{D}} = \Theta(t_{\mathtt{D}})$ table lookups. In comparison, the value $T_{\mathtt{H}}$ for the Hellman tradeoff corresponds to $T_{\mathtt{H}} = \Theta(t_{\mathtt{H}}^2)$ one-way function computations and, as testified through Lemma 28, table lookups of the same $\Theta(t_{\mathtt{H}}^2)$ order.

Hence, even if we are working with two parameter sets for the DP and Hellman tradeoffs which lead to identical time $T_{\mathtt{D}} = T_{\mathtt{H}}$, the real-world execution time of the two algorithms will be different. For a fair interpretation of the tradeoff coefficient ratio $\mathtt{D}_{tc} : \mathtt{H}_{tc}$ as a ratio of tradeoff performances, the difference in the time units used by the two algorithms must be taken into account.

To continue the discussion, we recall the online time complexity of the rainbow tradeoff. In this case, we can expect the appearance of the time value $T_{\mathtt{R}}$ to call for $T_{\mathtt{R}} = \Theta(t_{\mathtt{R}}^2 l_{\mathtt{R}})$ one-way function computations and, according to Lemma 34, table lookups of order $\Theta(t_{\mathtt{R}} l_{\mathtt{R}})$.

In both the DP and rainbow tradeoffs, the number of table lookups is of strictly smaller order than the number of one-way function computations. Hence, in these cases, we may ignore the time taken by table lookups and treat $T_{\mathtt{D}}$ and $T_{\mathtt{R}}$, the count of one-way function invocations, as the execution time. In the Hellman case, we can combine (15) and Lemma 28 to conclude that a $T_{\mathtt{H}}$ must be understood as $T_{\mathtt{H}}$ one-way function computations and $\frac{6}{6+\mathtt{H}_{msr}} T_{\mathtt{H}}$ table lookups.

Ignoring any issues concerning the storage count M for the moment, we can state that to compare the DP or rainbow tradeoff algorithm against the Hellman tradeoff, one should look at the ratios

$$\mathtt{D}_{tc} : \Bigl(1 + \frac{6}{6+\mathtt{H}_{msr}} \frac{\text{single table lookup time}}{\text{single one-way function computation time}}\Bigr)\mathtt{H}_{tc}$$

and

$$\Bigl(1 + \frac{6}{6+\mathtt{H}_{msr}} \frac{\text{single table lookup time}}{\text{single one-way function computation time}}\Bigr)\mathtt{H}_{tc} : \mathtt{R}_{tc}$$

rather than the raw ratios $\mathtt{D}_{tc} : \mathtt{H}_{tc}$ and $\mathtt{H}_{tc} : \mathtt{R}_{tc}$.

If the one-way function is computationally very heavy and all the pre-computed tables are to reside on the fast online memory during the online phase, then the table lookup time could be insignificant in comparison to the one-way function computation time. In such a case, the above ratios would essentially reduce to $\mathtt{D}_{tc} : \mathtt{H}_{tc}$ and $\mathtt{H}_{tc} : \mathtt{R}_{tc}$. On the other hand, if huge pre-computed tables are to be accessed through slow network storage and the one-way function is computationally very light, the above conversion of units will be necessary.

It is clear that when comparing the DP tradeoff against the rainbow tradeoff, no conversion of the time units is necessary. At least when the issue of the storage unit is ignored, the ratio $\mathtt{D}_{tc} : \mathtt{R}_{tc}$ is equal to the tradeoff performance ratio.

Unit for M. Let us now discuss the storage unit. In all three tradeoff algorithms, M represents the number of starting point and ending point pairs that need to be stored, but the actual number of bits required to store each table entry will be different for different tradeoff algorithms. We saw through Section 5.3 that, for the DP tradeoff, slightly more than $\log m_{\mathtt{D}}$ bits are required to store a single starting point and ending point pair. On the other hand, slightly more than $\log m_{\mathtt{H}} + \log t_{\mathtt{H}}$ bits are required for the Hellman tradeoff, and the rainbow tradeoff requires slightly more than $\log m_{\mathtt{R}}$ bits to store each table entry.

The implementation environment and tradeoff requirements will place the choice of suitable parameters into a certain range, and it is reasonable to assume that the parameters chosen to be used with each algorithm will be related by

$$\log m_{\mathtt{D}} \approx \log m_{\mathtt{H}}, \quad \log t_{\mathtt{D}} \approx \log t_{\mathtt{H}}, \quad \text{and} \quad \log m_{\mathtt{R}} \approx \log m_{\mathtt{D}} + \log t_{\mathtt{D}} \approx \log m_{\mathtt{H}} + \log t_{\mathtt{H}}.$$

Some readers may object that our discussion on the number of bits required for each table entry makes $m_{\mathtt{D}} = 2m_{\mathtt{H}}$ a more reasonable choice, but this difference by a factor two is lost in the approximation when, as in the above assumption, their logarithm values are compared.

Given the same amount of physical storage, the number of table entries that can be stored by the DP tradeoff will be greater than the number of entries that can be stored by the Hellman tradeoff by a factor of

$$\frac{\log m_{\mathtt{H}} + \log t_{\mathtt{H}}}{\log m_{\mathtt{D}}} \approx 1 + \frac{\log t_{\mathtt{D}}}{\log m_{\mathtt{D}}} \approx 1 + \frac{\log t_{\mathtt{H}}}{\log m_{\mathtt{H}}}.$$

Noting that the change in storage affects the tradeoff performance through a square factor, and ignoring effects of time unit differences for the moment, we can state that, to compensate for the storage unit differences, the ratio

$$\mathtt{D}_{tc} : \left(1+\frac{\log t_{\mathtt{H}}}{\log m_{\mathtt{H}}}\right)^2 \mathtt{H}_{tc}$$

should be used instead of $\mathtt{D}_{tc} : \mathtt{H}_{tc}$ for comparison of tradeoff performances. Similarly, ignoring the time unit, since we have $\frac{\log m_{\mathtt{R}}}{\log m_{\mathtt{D}}} \approx 1 + \frac{\log t_{\mathtt{D}}}{\log m_{\mathtt{D}}}$, the ratio $\mathtt{D}_{tc} : \mathtt{R}_{tc}$ should be converted into

$$\mathtt{D}_{tc} : \left(1+\frac{\log t_{\mathtt{D}}}{\log m_{\mathtt{D}}}\right)^2 \mathtt{R}_{tc}.$$

The remaining ratio $\mathtt{H}_{tc} : \mathtt{R}_{tc}$ requires no conversion to deal with storage units, since we have $\log m_{\mathtt{H}} + \log t_{\mathtt{H}} \approx \log m_{\mathtt{R}}$.

Combined unit conversion. It now suffices to combine the two arguments concerning units of time and storage to give fair comparisons of different tradeoff algorithms.

Proposition 40 *Consider different tradeoff algorithms that are set to run with specific corresponding parameters. Suppose that the tradeoff coefficients for the algorithms are given as* $\mathtt{D}_{tc}$, $\mathtt{H}_{tc}$, *and* $\mathtt{R}_{tc}$. *Then the tradeoff performance ratios between tradeoff algorithms are given by the ratios*

$$\mathtt{D}_{tc} : \left(1+\frac{6}{6+\mathtt{H}_{msr}}\,\frac{\text{single table lookup time}}{\text{single one-way function computation time}}\right)\left(1+\frac{\log t_{\mathtt{H}}}{\log m_{\mathtt{H}}}\right)^2 \mathtt{H}_{tc},$$

$$\mathtt{D}_{tc} : \left(1+\frac{\log t_{\mathtt{D}}}{\log m_{\mathtt{D}}}\right)^2 \mathtt{R}_{tc},$$

and

$$\left(1+\frac{6}{6+\mathtt{H}_{msr}}\,\frac{\text{single table lookup time}}{\text{single one-way function computation time}}\right)\mathtt{H}_{tc} : \mathtt{R}_{tc}.$$

In our further discussions below, we shall mainly work with parameter sets that roughly satisfy

$$\log m_{\mathtt{D}} \approx \log m_{\mathtt{H}} \approx \log t_{\mathtt{D}} \approx \log t_{\mathtt{H}} \approx \log t_{\mathtt{R}} \approx \frac{1}{3}\log \mathsf{N} \quad \text{and} \quad \log m_{\mathtt{R}} \approx \frac{2}{3}\log \mathsf{N}.$$

and also mostly assume that the table lookup time is negligible in comparison to the one-way function computation time. Under these assumptions, the ratios that need to be studied are

$$\mathtt{D}_{tc} : 4\mathtt{H}_{tc}, \quad \mathtt{D}_{tc} : 4\mathtt{R}_{tc}, \quad \text{and} \quad \mathtt{H}_{tc} : \mathtt{R}_{tc}.$$

Hence, it suffices to compare the values $\frac{1}{4}\mathtt{D}_{tc}$, $\mathtt{H}_{tc}$, and $\mathtt{R}_{tc}$ against each other.

We shall refer to the above situation as the *typical situation*, as it often appears during theoretic developments of the tradeoff technique, but we do not claim this to be typical in practical applications of the tradeoff technique.

We emphasize that our further discussions given below concerning tradeoff performance comparisons will only be valid under the typical environment assumption that was just explained. If the environment and tradeoff performance requirements make parameter choices such that $m_{\mathtt{H}} \gg t_{\mathtt{H}}$ or $m_{\mathtt{H}} \ll t_{\mathtt{H}}$ more appropriate, or if the table lookup time is not negligible in comparison to the one-way function computation time, the conclusions will be different. Still, one will be able to start from Proposition 40 and similarly discuss these other situations.

9.2 DP tradeoff versus Hellman tradeoff

The focus of this work is with practical uses of the tradeoff algorithms, and we shall restrict discussion of the DP tradeoff to the case when $\hat{t} \gg t$. As discussed in the previous subsection, it suffices to compare $\frac{1}{4}\mathtt{D}_{tc}$ against $\mathtt{H}_{tc}$ for a fair comparison between the DP and Hellman tradeoffs. Note that we are assuming the typical situation explained at the end of the previous subsection and any conclusion we make could be different under different circumstances.

The contents of Proposition 37 and Proposition 38 show that the optimal performances of the two algorithms are given by

$$\frac{1}{4}\mathtt{D}_{tc} = 1.75264\,\mathtt{D}_{ps}\big\{\ln(1-\mathtt{D}_{ps})\big\}^2 \quad \text{and} \quad \mathtt{H}_{tc} = 1.50217\,\mathtt{H}_{ps}\big\{\ln(1-\mathtt{H}_{ps})\big\}^2.$$

One may be lead to believe that the Hellman tradeoff, with the smaller tradeoff coefficient, will be more efficient, but this is only true when the pre-computation cost is totally ignored. In practice, pre-computation cost is the largest barrier to any large scale deployment of tradeoff algorithms and is hard to ignore.

The pre-computation costs required to achieve the above tradeoff performances are

$$\mathtt{D}_{pc} = 1.22871\,\ln\Big(\frac{1}{1-\mathtt{D}_{ps}}\Big) \quad \text{and} \quad \mathtt{H}_{pc} = 1.35021\,\ln\Big(\frac{1}{1-\mathtt{H}_{ps}}\Big).$$

The pre-computation cost of the DP tradeoff is seen to be lower and we are faced with the problem of how to compare low tradeoff performance at lower pre-computation against better tradeoff performance at higher pre-computation cost.

After a moment of thought one must admit that such a comparison can not be objective. It is closely related to the relative value of the tradeoff performance against the pre-computation effort, and there is no unit with which to express either of these values. As an extension of this thought, one must question whether it is reasonable to compare the two tradeoffs at parameters giving their respective optimal tradeoff performances. Non-optimal parameters may be preferable under many situations in view of lower pre-computation cost.

We can conclude that all we can do is present the range of choices that can be made and allow the users to make their conclusions based on their situation. The crucial information that must be displayed to allows easy judgement of which tradeoff is more suitable is the relation between tradeoff performance and pre-computation cost. This must be done under each fixed requirement for password recovery rate.

As was previously noted through (23) and (22), when under a fixed probability of success requirement, both $\mathtt{D}_{pc}$ and $\mathtt{H}_{pc}$ are functions of their respective $\mathtt{D}_{msr}$ and $\mathtt{H}_{msr}$ values. The tradeoff coefficients $\mathtt{D}_{tc}$ and $\mathtt{H}_{tc}$, under a fixed success rate requirement, were similarly expressed as functions of the corresponding $\mathtt{D}_{msr}$ and $\mathtt{H}_{msr}$ values in (19) and (20).

For a comparison of the DP tradeoff against the Hellman tradeoff, it now suffices to present the graphs

$$\Big\{\Big(\mathtt{D}_{pc}[\mathtt{D}_{msr}], \frac{1}{4}\mathtt{D}_{tc}[\mathtt{D}_{msr}]\Big) \mid \mathtt{D}_{msr} \le 0.562047\Big\} \tag{28}$$

and

$$\big\{\big(\mathtt{H}_{pc}[\mathtt{H}_{msr}], \mathtt{H}_{tc}[\mathtt{H}_{msr}]\big) \mid \mathtt{H}_{msr} \le 2.25433\big\}, \tag{29}$$

where the bounds on $\mathtt{D}_{msr}$ and $\mathtt{H}_{msr}$ were placed in accordance to the discussion at the end of Section 8.2. These graphs are given in Figure 5.

Since the two graphs are to be compared at identical password recovery rate requirements $\mathtt{D}_{ps} = \mathtt{H}_{ps}$, we have removed the common parts that depend on the success probability

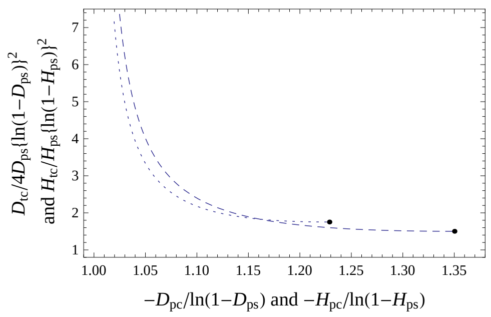


Fig. 5 The tradeoff coefficient $\frac{1}{4}\mathtt{D}_{tc}$ (dotted) and $\mathtt{H}_{tc}$ (dashed) in relation to their respective pre-computation cost

from both of the cases before plotting the graphs. Hence, the graphs do not depend on the success rate and are valid for all success rate requirements. Both graphs extend further upwards, but the right ends, corresponding to the optimal tradeoff performances, are clearly marked with dots.

The two graphs are very close to each other. Even though slightly better tradeoff performance can be obtained with the Hellman tradeoff at higher pre-computation cost, in practice, unless parameters far from the typical $m \approx t \approx \mathsf{N}^{\frac{1}{3}}$ region are to be used, the DP tradeoff will be favored in view of less number of table lookups. For example, if the table lookup time makes $\frac{1}{5}\mathtt{D}_{tc} : \mathtt{H}_{tc}$ a more appropriate measure of tradeoff performance ratio than the current $\frac{1}{4}\mathtt{D}_{tc} : \mathtt{H}_{tc}$, the dotted curve for the DP tradeoff would move down and present itself as a more advantageous algorithm.

If table lookup time is absolutely negligible in comparison to the one-way function computation time, there is a short range of parameter sets with which the Hellman tradeoff can slightly outperform the DP tradeoff using the same amount of pre-computation. If table lookup time is negligible and pre-computation is not to be considered, the Hellman tradeoff can be somewhat better.

9.3 Rainbow tradeoff versus DP and Hellman tradeoffs

As was discussed in Section 9.1, we assume the typical situation concerning the approximate range of parameters and table lookup time, and consider comparisons between $\frac{1}{4}\mathtt{D}_{tc}$, $\mathtt{H}_{tc}$, and $\mathtt{R}_{tc}$ to be fair.

In addition to the graphs (28) and (29), we need to plot all possible $(\mathtt{R}_{pc}, \mathtt{R}_{tc})$ points. We can first check through (26) that $\mathtt{R}_{pc}$ can be seen as a function of the table count l, when success rate requirement $\mathtt{R}_{ps}$ is fixed. As for the tradeoff coefficient, equation (25) presents it as a function of just l, when $\mathtt{R}_{ps}$ is fixed. Given any requirement on the success rate $\mathtt{R}_{ps}$, it is now possible to draw the graph

$$\{(\mathtt{R}_{pc}[l], \mathtt{R}_{tc}[l]) \mid l \geq \text{optimal table count for } \mathtt{R}_{ps}\}, \tag{30}$$

where the optimal table count can be obtained from Table 1. Note that this is no longer a continuous graph, but a discrete set of points. In the strict sense, previous graphs were also discrete set of points, but when N is large, the points will be extremely close to each other.

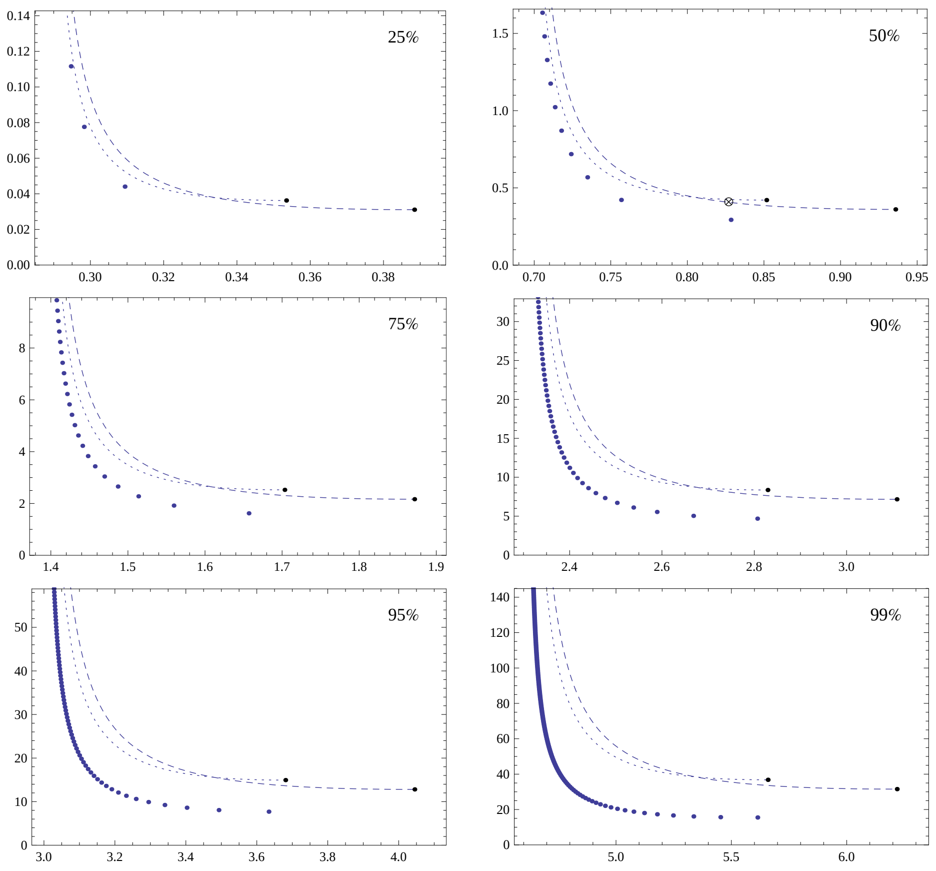


Fig. 6 Tradeoff coefficient $\frac{1}{4}\mathtt{D}_{tc}$ (dotted), $\mathtt{H}_{tc}$ (dashed), and $\mathtt{R}_{tc}$ (large dots) in relation to their respective pre-computation cost at success rates 25%, 50%, 75%, 90%, 95%, and 99% (X-axis: $\mathtt{D}_{pc}$, $\mathtt{H}_{pc}$, and $\mathtt{R}_{pc}$; Y-axis: $\frac{1}{4}\mathtt{D}_{tc}$, $\mathtt{H}_{tc}$, and $\mathtt{R}_{tc}$)

Unlike our comparison between DP and Hellman tradeoffs, the parts that depend on $\mathtt{R}_{ps}$ appearing in the expressions (26) and (25) are not identical to those appearing in the corresponding expressions (23), (22), (19), and (20). Hence separate graphs need to be drawn for each success rate. This is given in Figure 6 for some success rates.

In all of the graphs, one can see that the curve for the rainbow tradeoff sits closer to the origin than the curves for DP and Hellman tradeoffs. Note that a graph sitting lower shows better tradeoff performance and being positioned more to the left implies lower pre-computation cost. In all the cases except for the ones corresponding to 25% and 50% success rates, given any position on the curve for either the DP or Hellman tradeoff there is a rainbow tradeoff position that presents better tradeoff performance at a smaller pre-computation cost. Hence use of the rainbow tradeoff is definitely advisable in these cases.

The existence of better rainbow position is also mostly true in the 50% case. The exception is marked with an $\otimes$ on the curve for the Hellman tradeoff. This position is very slightly to the left of the optimal rainbow position and hence corresponds to less pre-computation than the optimal rainbow position. At the same time, it is positioned lower than the second best rainbow position and hence shows better tradeoff performance than this second best position. Hence, there can be no rainbow tradeoff parameter set that can replace the Hellman position marked with an $\otimes$ without at least very slightly sacrificing either the pre-computation cost or the tradeoff efficiency. Still, anybody can agree that this exception is quite unreasonable and one would normally choose to sacrifice the extremely small amount of either the pre-computation cost or the tradeoff performance for a somewhat better value of the other factor.

The 25% case also displays the rainbow tradeoff requiring less pre-computation than the other two tradeoffs in achieving the equal tradeoff performance, but the awkward exceptional position discussed for the 50% can be found here as rather large segments. In addition, the best performance achievable by the rainbow tradeoff falls short of what is reachable by the other two algorithms. Hence there may be situations where the DP or Hellman tradeoffs may be preferable over the rainbow tradeoff, when required to achieve 25% success rate.

The relative advantage of using rainbow tradeoff is clearly seen to grow with the increase in the success rate requirement. For the 99% success rate case, it seems almost safe to say that the rainbow tradeoff is approximately two times more efficient than the other two tradeoffs in any of their reasonable usages.

In conclusion, the use of rainbow tradeoff is advisable for high success rate requirements and there may occasionally be low success rate applications with special situations where the other two tradeoffs are preferable. We emphasize once more that this conclusion is only valid under the typical situation assumption explained in Section 9.1. For example, if we must work with parameters such that $2\log m_{\mathtt{D}} \approx \log t_{\mathtt{D}}$ and $2\log m_{\mathtt{H}} \approx \log t_{\mathtt{H}}$, then comparison of the coefficients $\frac{1}{9}\mathtt{D}_{tc}$, $\mathtt{H}_{tc}$, and $\mathtt{R}_{tc}$ would be appropriate, which would bring the curve for the DP tradeoff lower, leading to different conclusions.

10 Conclusion

In the first part of this work, we solidified the basis on which analysis of tradeoff algorithms may be discussed. A logical gap in common arguments involving random functions was identified and plausible explanation for ignoring the problem was given. We next studied the performance of DP, Hellman, and rainbow tradeoffs, and summarized each as a tradeoff curve that is correct even to the small multiplicative factor. These results were used in the last part of this work to compare the performance of tradeoff algorithms against each other.

Although we did provide explicit statements comparing the three tradeoff algorithms, our conclusions are only true under a certain assumption on the tradeoff situation. We emphasize once more that one should not extend our conclusions to other situations. Rather, one should see this work as providing the tools that allow for a fair comparison of tradeoff algorithms and use these to arrive at their own final judgements.

One conclusion we can provide about the relative performance of different tradeoff algorithms is that any difference in performance will be rather small. The practical inconvenience of having to align each entry of the pre-computed tradeoff table at a byte boundary has not been considered in this work, and the performance differences between algorithms can be so small that such obscure issues may be of more importance in practice.

The fact that algorithm performances are not very different is disappointing to us as authors of this work, but this fact should be relieving to practitioners of the tradeoff algorithm that are not concerned with small performance differences. Nevertheless, even if one decides to ignore small performance differences, graphs of the previous section show that meaningful reduction in pre-computation cost can be achieved with only a small sacrifice to tradeoff performance and being able to take advantage of this knowledge will be of practical importance.

This work did not consider the use of checkpoints [1], which can be used to reduce the cost of false alarms. This decision was mostly based on the work [9], where the effect of checkpoints in reducing the online time of Hellman and non-perfect rainbow tradeoffs was shown to be quite smaller than 10% at typical parameters. Since checkpoints will affect both algorithms in a positive way, its effect on the final comparison of algorithms will be minimal. Therefore, we chose not to take the checkpoint technique into account, considering the complications to the analysis its introduction would bring. Still, the effect of checkpoints on the DP tradeoff has not yet been accurately analyzed and there is a small possibility that its behavior on the DP tradeoff will be different from those on the other algorithms.

Analysis of perfect table versions of the tradeoff algorithms analogous to what is given here also remains to be done, with some partial results available from [1, 9, 15]. Due to the larger pre-computation cost, the perfect table cases may be of less practical interest, but they are certainly very interesting from a theoretical viewpoint.

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 in INDOCRYPT 2005
2. E. Barkan, E. Biham, A. Shamir, Rigorous bounds on cryptanalytic time/memory tradeoffs, in *Advances in Cryptology—CRYPTO 2006*, LNCS, vol. 4117, (Springer, 2006), pp .1–21
3. A. Biryukov, A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers, in *Advances in Cryptology—ASIACRYPT 2000*, LNCS, vol. 1976, (Springer, 2000), pp. 1–13
4. D. E. Denning, *Cryptography and Data Security* (Addison-Wesley, 1982)
5. P. Flajolet, A. M. Odlyzko, Random mapping statistics, in *Advances in Cryptology—EUROCRYPT '89*, LNCS, vol. 434, (Springer, 1990), pp. 329–354
6. S. Goldwasser, M. Bellare, Leture Notes on Cryptography. Unpublished manuscript, July 2008. Available at: `http://cseweb.ucsd.edu/~mihir/papers/gb.html`
7. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher, in *Advances in Cryptology—EUROCRYPT '97*, LNCS, vol. 1233, (Springer, 1997), pp. 239–255
8. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory*, **26**, 401–406 (1980)
9. J. Hong, The cost of false alarms in Hellman and rainbow tradeoffs. *Des. Codes Cryptogr.*, to appear
10. D. Ma, J. Hong, Success probability of the Hellman trade-off. *Inf. Process. Lett.*, **109**(7), 347–351 (2009)
11. A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, 1997)
12. S. Moon, Parameter selection in cryptanalytic time memory tradeoffs. MS Thesis, Seoul National University, June 2009.
13. P. Oechslin, Making a faster cryptanalytic time-memory trade-off. in *Advances in Cryptology–CRYPTO 2003*, LNCS, vol. 2729, (Springer, 2003) pp .617–630
14. C. Schnorr, H. Lenstra, Jr., A Monte Carlo factoring algorithm with linear storage, in *Math Comp.*, Vol. 43(167), pp.289–311, 1984
15. 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, vol. 2523, (Springer, 2003), pp. 593–609

A Approximation $\exp(\mathsf{A}/\mathsf{B}) \approx (1-1/\mathsf{A})^{\mathsf{B}}$

In this section, we show that

$$\left|\exp\left(-\frac{\mathsf{A}}{\mathsf{B}}\right) - \left(1-\frac{1}{\mathsf{B}}\right)^{\mathsf{A}}\right| < \left\{\frac{1}{2}\frac{\mathsf{A}}{\mathsf{B}^2} + \frac{1}{(\mathsf{A}+1)!}\left(\frac{\mathsf{A}}{\mathsf{B}}\right)^{\mathsf{A}+1}\right\}\exp\left(\frac{\mathsf{A}}{\mathsf{B}}\right) \tag{31}$$

for all positive integers A and B. This is the content of Lemma 1.

We start by writing $\exp\big(-\frac{\mathsf{A}}{\mathsf{B}}\big)$ in its Taylor series form and fully expanding the term $(1-\frac{1}{\mathsf{B}})^{\mathsf{A}}$.

$$\begin{aligned}
&\Big|\exp\Big(-\frac{\mathsf{A}}{\mathsf{B}}\Big)-\Big(1-\frac{1}{\mathsf{B}}\Big)^{\mathsf{A}}\Big| \\
&=\Big|\Big\{1-\frac{\mathsf{A}}{\mathsf{B}}+\frac{1}{2!}\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)^2-\cdots\Big\}-\Big\{1-\binom{\mathsf{A}}{1}\frac{1}{\mathsf{B}}+\binom{\mathsf{A}}{2}\frac{1}{\mathsf{B}^2}-\cdots+(-1)^{\mathsf{A}}\binom{\mathsf{A}}{\mathsf{A}}\frac{1}{\mathsf{B}^{\mathsf{A}}}\Big\}\Big|.
\end{aligned}$$

After noting that the beginning two pairs of terms cancel out, we collect corresponding pairs from the two sequences of terms and bound the above by

$$\Big\{\Big|\frac{\mathsf{A}^2}{2!}-\binom{\mathsf{A}}{2}\Big|\frac{1}{\mathsf{B}^2}+\cdots+\Big|\frac{\mathsf{A}^{\mathsf{A}}}{\mathsf{A}!}-\binom{\mathsf{A}}{\mathsf{A}}\Big|\frac{1}{\mathsf{B}^{\mathsf{A}}}\Big\}+\Big\{\frac{1}{(\mathsf{A}+1)!}\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)^{\mathsf{A}+1}+\cdots\Big\}. \tag{32}$$

It is easy to see that

$$\begin{aligned}
0\le\frac{\mathsf{A}^k}{k!}-\binom{\mathsf{A}}{k}&=\frac{1}{k!}\big\{\mathsf{A}^k-\mathsf{A}(\mathsf{A}-1)\cdots(\mathsf{A}-k+1)\big\}\\
&=\frac{1}{k!}\Big\{\frac{k(k-1)}{2}\mathsf{A}^{k-1}-\cdots+(-1)^k(k-1)!\,\mathsf{A}\Big\}\\
&\le\frac{1}{k!}\frac{k(k-1)}{2}\mathsf{A}^{k-1}=\frac{1}{2}\frac{\mathsf{A}^{k-1}}{(k-2)!},
\end{aligned}$$

for every $k\ge2$, where the last inequality can be checked through induction on k. This shows that the terms of (32) that appear inside the first set of braces is bounded by

$$\begin{aligned}
&\frac{1}{2}\Big\{\frac{\mathsf{A}}{0!}\frac{1}{\mathsf{B}^2}+\frac{\mathsf{A}^2}{1!}\frac{1}{\mathsf{B}^3}+\frac{\mathsf{A}^3}{2!}\frac{1}{\mathsf{B}^4}+\cdots+\frac{\mathsf{A}^{\mathsf{A}-1}}{(\mathsf{A}-2)!}\frac{1}{\mathsf{B}^{\mathsf{A}}}\Big\}\\
&\quad=\frac{1}{2}\frac{\mathsf{A}}{\mathsf{B}^2}\Big\{1+\frac{1}{1!}\frac{\mathsf{A}}{\mathsf{B}}+\frac{1}{2!}\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)^2+\cdots+\frac{1}{(\mathsf{A}-2)!}\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)^{\mathsf{A}-2}\Big\}\\
&\quad\le\frac{1}{2}\frac{\mathsf{A}}{\mathsf{B}^2}\exp\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big).
\end{aligned}$$

As for the second set of braces from (32), it is easy to see that

$$\frac{1}{(\mathsf{A}+1)!}\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)^{\mathsf{A}+1}\exp\Big(\frac{\mathsf{A}}{\mathsf{B}}\Big)$$

can serve as its very rough bound. It now suffices to gather the two bounds to arrive at the claim (31).

B Standard Deviation of Image Sizes

The purpose of the section is to provide a proof to Lemma 7 concerning the standard deviation of image sizes. We first prepare a couple of technical lemmas for later use.

Lemma 41 *Let $F:\mathscr{N}\to\mathscr{N}$ be the random function. Fix a subset $\mathscr{M}\subset\mathscr{N}$ of size m and let $\mathbf{y}_1,\mathbf{y}_2\in\mathscr{N}$ be any two distinct points. The probability for $F(\mathscr{M})$ to contain both $\mathbf{y}_1$ and $\mathbf{y}_2$ is*

$$\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\}^2-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\{\Big(1-\frac{1}{\mathsf{N}}\Big)^m-\Big(1-\frac{1}{\mathsf{N}-1}\Big)^m\Big\}.$$

Proof The probability under consideration may be computed as follows.

$$\begin{aligned}
&\binom{m}{1}\Big(\frac{1}{\mathsf{N}}\Big)^1\Big(1-\frac{1}{\mathsf{N}}\Big)^{m-1}\Big\{1-\Big(1-\frac{1}{\mathsf{N}-1}\Big)^{m-1}\Big\}\\
&+\binom{m}{2}\Big(\frac{1}{\mathsf{N}}\Big)^2\Big(1-\frac{1}{\mathsf{N}}\Big)^{m-2}\Big\{1-\Big(1-\frac{1}{\mathsf{N}-1}\Big)^{m-2}\Big\}\\
&+\cdots\\
&+\binom{m}{m-1}\Big(\frac{1}{\mathsf{N}}\Big)^{m-1}\Big(1-\frac{1}{\mathsf{N}}\Big)^1\Big\{1-\Big(1-\frac{1}{\mathsf{N}-1}\Big)^1\Big\}
\end{aligned}$$

In each additive term, the part $\binom{m}{k}\left(\frac{1}{\mathsf{N}}\right)^k\left(1-\frac{1}{\mathsf{N}}\right)^{m-k}$ gives the probability for exactly k out of the m inputs to map to $\mathbf{y}_1$. The remaining $\left\{1-\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-k}\right\}$ part is the probability for at least one of the $(m-k)$ inputs that are known not to have reached $\mathbf{y}_1$ to map to $\mathbf{y}_2$.

The above sum is equal to the expression

$$\left\{\frac{1}{\mathsf{N}}+\left(1-\frac{1}{\mathsf{N}}\right)\right\}^m-\left\{\frac{1}{\mathsf{N}}+\left(1-\frac{1}{\mathsf{N}}\right)\left(1-\frac{1}{\mathsf{N}-1}\right)\right\}^m-\left(1-\frac{1}{\mathsf{N}}\right)^m+\left(1-\frac{1}{\mathsf{N}}\right)^m\left(1-\frac{1}{\mathsf{N}-1}\right)^m.$$

To check this claim, it suffices to expand the first two pairs of braces. The above expression can be rewritten in the form stated by this lemma. □

Lemma 42 *When* $1 \le m \le \mathsf{N}$, *we have the following asymptotic behaviors as* N *is increased.*

$$\left(1-\frac{1}{\mathsf{N}}\right)^m = \Theta(1).$$
$$1-\left(1-\frac{1}{\mathsf{N}}\right)^m = \Theta\left(\frac{m}{\mathsf{N}}\right).$$
$$\left(1-\frac{1}{\mathsf{N}}\right)^m-\left(1-\frac{1}{\mathsf{N}-1}\right)^m = \Theta\left(\frac{m}{\mathsf{N}^2}\right).$$

Proof The approximation $\left(1-\frac{1}{\mathsf{N}}\right)^m \approx \exp\left(-\frac{m}{\mathsf{N}}\right)$ is valid for all large N, while the condition $1 \le m \le \mathsf{N}$ implies $\frac{1}{e} \le \exp\left(-\frac{m}{\mathsf{N}}\right) < 1$ and shows that $\exp\left(-\frac{m}{\mathsf{N}}\right) = \Theta(1)$. This proves the first statement.

The following can be stated concerning the second claim, where the approximation is valid for all large N.

$$1-\left(1-\frac{1}{\mathsf{N}}\right)^m \;\approx\; 1-\exp\left(-\frac{m}{\mathsf{N}}\right) \;=\; \frac{m}{\mathsf{N}}-\frac{1}{2!}\left(\frac{m}{\mathsf{N}}\right)^2+\frac{1}{3!}\left(\frac{m}{\mathsf{N}}\right)^3-\cdots.$$

If m is strictly less than N, we may claim that the righthand side end is $\Theta\left(\frac{m}{\mathsf{N}}\right)$. If $m = \mathsf{N}$, the above is $\frac{1}{e}$ which we may again claim to be $\Theta\left(\frac{m}{\mathsf{N}}\right)$.

As for the final statement, we can check that

$$\begin{aligned}
&\left(1-\frac{1}{\mathsf{N}-1}\right)^m-\left(1-\frac{1}{\mathsf{N}}\right)^m \\
&\quad=\left\{\left(1-\frac{1}{\mathsf{N}-1}\right)-\left(1-\frac{1}{\mathsf{N}}\right)\right\}\left\{\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-1}+\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-2}\left(1-\frac{1}{\mathsf{N}}\right)+\cdots+\left(1-\frac{1}{\mathsf{N}}\right)^{m-1}\right\} \\
&\quad=\frac{1}{\mathsf{N}(\mathsf{N}-1)}\left\{\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-1}+\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-2}\left(1-\frac{1}{\mathsf{N}}\right)+\cdots+\left(1-\frac{1}{\mathsf{N}}\right)^{m-1}\right\}.
\end{aligned}$$

This shows that

$$\frac{1}{\mathsf{N}(\mathsf{N}-1)}\,m\left(1-\frac{1}{\mathsf{N}-1}\right)^{m-1} \le \left(1-\frac{1}{\mathsf{N}-1}\right)^m-\left(1-\frac{1}{\mathsf{N}}\right)^m \le \frac{1}{\mathsf{N}(\mathsf{N}-1)}\,m\left(1-\frac{1}{\mathsf{N}}\right)^{m-1}$$

and we can claim $\Theta\left(\frac{m}{\mathsf{N}^2}\right)$ order for the middle quantity. This concludes the proof. □

In the remainder of this section, $\mathscr{M}$ will be a fixed subset of $\mathscr{N}$ that is of size m. For each $\mathbf{y} \in \mathscr{N}$, let us define the function $\chi_{\mathbf{y}} : \mathscr{N}^{\mathscr{N}} \to \{0,1\}$ by

$$\chi_{\mathbf{y}}(F) = \begin{cases} 0 & \text{if } \mathbf{y} \in F(\mathscr{M}), \\ 1 & \text{if } \mathbf{y} \notin F(\mathscr{M}). \end{cases}$$

The dependence of $\chi_{\mathbf{y}}$ on $\mathscr{M}$ was not made explicit in the notation since we will keep $\mathscr{M}$ fixed throughout this section. The size of the image of $\mathscr{M}$ under any function $F : \mathscr{N} \to \mathscr{N}$ can be expressed in terms of this indicator function as

$$|F(\mathscr{M})| = \sum_{\mathbf{y}\in\mathscr{N}} \chi_{\mathbf{y}}(F).$$

Using this observation, one can present

$$E\left[|F(\mathscr{M})|\right] = E\Big[\sum_{\mathbf{y}\in\mathscr{N}} \chi_{\mathbf{y}}(F)\Big] = \sum_{\mathbf{y}\in\mathscr{N}} E\left[\chi_{\mathbf{y}}(F)\right] = \mathsf{N}\,E\left[\chi_{\mathbf{y}'}(F)\right] = \mathsf{N}\left\{1-\left(1-\frac{1}{\mathsf{N}}\right)^m\right\}, \tag{33}$$

where $\mathbf{y}'$ is any fixed point of $\mathcal{N}$, as an alternative way of writing down the proof to Lemma 3.

Let us fix the notation

$$\chi = \sum_{\mathbf{y}\in\mathcal{N}} \chi_{\mathbf{y}}$$

and view this as a random variable defined on the space $\mathcal{N}^{\mathcal{N}}$, which is given the uniform probability distribution. It maps each element F to the positive integer $|F(\mathcal{M})|$. Equation (33) is equivalent to

$$E[\chi] = \mathsf{N}\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\} \tag{34}$$

and the goal of this section is to show that the standard deviation

$$\mathrm{stdev}(\chi) = \sqrt{E[\chi^2]-(E[\chi])^2}$$

is of $O(\sqrt{m})$ order.

One can easily check that

$$\begin{aligned} E[\chi^2] &= E\Big[\big(\sum_{\mathbf{y}}\chi_{\mathbf{y}}\big)^2\Big] = E\Big[\sum_{\mathbf{y}_1,\mathbf{y}_2}\chi_{\mathbf{y}_1}\chi_{\mathbf{y}_2}\Big] = E\Big[\sum_{\mathbf{y}}\chi_{\mathbf{y}} + \sum_{\mathbf{y}_1\neq\mathbf{y}_2}\chi_{\mathbf{y}_1}\chi_{\mathbf{y}_2}\Big] = E[\chi] + \sum_{\mathbf{y}_1\neq\mathbf{y}_2} E\big[\chi_{\mathbf{y}_1}\chi_{\mathbf{y}_2}\big] \\ &= E[\chi] + \mathsf{N}(\mathsf{N}-1)E\big[\chi_{\mathbf{y}'_1}\chi_{\mathbf{y}'_2}\big], \end{aligned}$$

where $\mathbf{y}'_1$ and $\mathbf{y}'_2$ are any two distinct points of $\mathcal{N}$. The expectation $E\big[\chi_{\mathbf{y}'_1}\chi_{\mathbf{y}'_2}\big]$ is equal to the probability for both $\mathbf{y}'_1$ and $\mathbf{y}'_2$ to belong to the image space, and this is the content of Lemma 41. Referring also to (34) and Lemma 42, we can compute the variance as follows.

$$\begin{aligned} \big\{\mathrm{stdev}(\chi)\big\}^2 &= E[\chi^2]-(E[\chi])^2 \\ &= E[\chi] + \mathsf{N}(\mathsf{N}-1)E\big[\chi_{\mathbf{y}'_1}\chi_{\mathbf{y}'_2}\big] - (E[\chi])^2 \\ &= \mathsf{N}\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\} + \mathsf{N}(\mathsf{N}-1)\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\}^2 \\ &\quad\; -\mathsf{N}(\mathsf{N}-1)\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\{\Big(1-\frac{1}{\mathsf{N}}\Big)^m-\Big(1-\frac{1}{\mathsf{N}-1}\Big)^m\Big\} - \mathsf{N}^2\Big\{1-\Big(1-\frac{1}{\mathsf{N}}\Big)^m\Big\}^2 \\ &= \mathsf{N}\,\Theta\Big(\frac{m}{\mathsf{N}}\Big) - \mathsf{N}\,\Theta\Big(\frac{m}{\mathsf{N}}\Big)^2 - \mathsf{N}(\mathsf{N}-1)\,\Theta(1)\,\Theta\Big(\frac{m}{\mathsf{N}^2}\Big) \\ &= \Theta(m) - \Theta(m)\,O(1) - \Theta(m) \end{aligned}$$

Finally, since the last expression is clearly of $O(m)$ order, we can claim that $\mathrm{stdev}(\chi) = O(\sqrt{m})$. This concludes the proof of Lemma 7.

C Experiment Results

In this section we verify that the main parts of our arguments agree well with experiment results. Experiments are done to check the validity of our results concerning the coverage rate and the cost of false alarms for the DP tradeoff. Analogous testings for the Hellman and rainbow tradeoffs are not provided, as these testings were done in [9]. We also provide experimental evidence supporting our arguments surrounding the effects of the ending point truncation method.

Since averaging over all functions defined on any reasonably large space is not possible, all our tests were conducted with a very small subset of explicitly constructed one-way functions. The one-way function used was always the encryption key to ciphertext mapping, under a fixed plaintext, computed with the blockcipher AES-128. Different randomly chosen plaintexts were used to provide multiple one-way functions. The size of the input space was controlled by utilizing only a small number of key bits and padding the remaining key bits with zeros. The output space size was controlled by masking the ciphertext to an appropriate bit length. When working with the DP tradeoff, as discussed at the start of Section 5, we initially constructed $m_0 = \frac{m}{1-e^{-\hat{t}/t}}$ precomputation chains and gathered every resulting DP chain, rather than incrementally generate additional chains until m DP chains were collected.

Table 2 Coverage rate of DP tradeoff ($\mathsf{N} = 2^{30}$)

$\log m$	$\log t$	$\hat{t}/t$	$\mathtt{D}_{msr}$	test	theory
11	9	1/2	0.5	0.225357	0.224285
9	10	1/2	0.5	0.225368	0.224285
11	9	1	0.5	0.400071	0.399566
9	10	1	0.5	0.398824	0.399566
11	9	2	0.5	0.628349	0.627405
9	10	2	0.5	0.629802	0.627405
11	9	5	0.5	0.816415	0.814339
9	10	5	0.5	0.811530	0.814339
12	9	1/2	1.0	0.221190	0.219643
10	10	1/2	1.0	0.220655	0.219643
12	9	1	1.0	0.384839	0.383464
10	10	1	1.0	0.385049	0.383464
12	9	2	1.0	0.582370	0.581801
10	10	2	1.0	0.581019	0.581801
12	9	5	1.0	0.722192	0.723263
10	10	5	1.0	0.721465	0.723263
13	9	1/2	2.0	0.212476	0.211204
11	10	1/2	2.0	0.212538	0.211204
13	9	1	2.0	0.357424	0.356587
11	10	1	2.0	0.355287	0.356587
13	9	2	2.0	0.515214	0.515495
11	10	2	2.0	0.514631	0.515495
13	9	5	2.0	0.611834	0.612748
11	10	5	2.0	0.610616	0.612748

C.1 Coverage rate of DP tradeoffs

Experiment results supporting Proposition 17, which presents the coverage rate of a DP table, are given in Table 2. The coverage rate was measured by simply storing all DP matrix entries while constructing the DP chains and later counting the number of distinct matrix entries that were used as inputs to the one-way function. Each test result value given in the table is an average over 100 experiments. Different randomly generated plaintexts were used for each of these experiment. All the tests were done on a space of size $\mathsf{N} = 2^{30}$. One can check that the test figures are very close to what the theory predicted.

C.2 False alarm cost of DP tradeoffs

Our next goal is to check the validity of our arguments concerning the time complexity that incorporates the extra cost of false alarms. We could do this with the expression for time complexity stated during the proof of Theorem 21, but such an approach could hide much of the inner workings. Hence, we decided to verify the following lemma, which allows access to much finer detail.

Lemma 43 *Consider the DP tradeoff. The expected number of chain collisions at the i-th iteration of the online phase is*

$$\frac{1}{t}\,\frac{\mathtt{D}_{msr}}{1-e^{-\hat{t}/t}}\left\{-e^{-\hat{t}/t}+e^{-\hat{t}/t}\exp\Big(-\frac{i}{t}\Big)+\frac{i}{t}\,\exp\Big(-\frac{i}{t}\Big)\right\}.$$

Proof The expected number of chain collisions is the sum over all rows of the DP matrix of the respective probabilities for the i-th iteration to sound an alarm in association with that row. After reading the proof to Lemma 20, it should be clear that the sum of probabilities we are looking for is

$$\sum_{j=1}^{\hat{t}}\frac{\frac{m}{t}}{1-e^{-\hat{t}/t}}\,\exp\Big(-\frac{j}{t}\Big)\cdot\frac{t}{\mathsf{N}}\Big\{\exp\Big(\frac{\min\{i,j\}}{t}\Big)-1\Big\}\exp\Big(-\frac{i}{t}\Big).$$

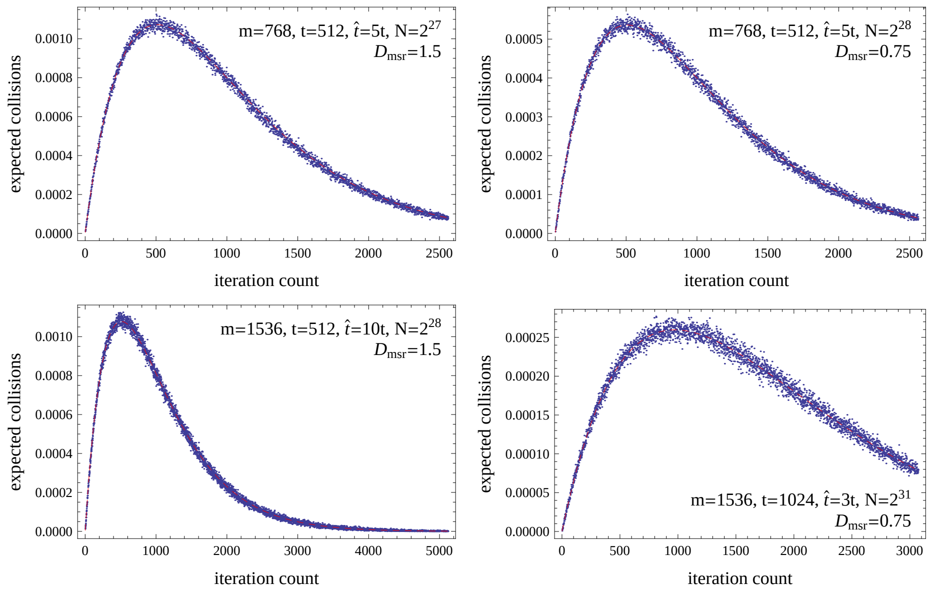


Fig. 7 Expected number of collisions at each iteration of the DP tradeoff (dots: experiment; dashed line: theory)

In integral form, this is approximately

$$\frac{1}{t}\,\frac{\frac{mt^2}{\mathsf{N}}}{1-e^{-\hat{t}/t}}\,\exp\left(-\frac{i}{t}\right)\int_0^{\hat{t}/t}\exp(-v)\left\{\exp\left(\min\left\{\frac{i}{t},v\right\}\right)-1\right\}dv,$$

which simplifies to what is claimed. □

This lemma contains the core of our arguments given in the main text concerning the cost of alarms, and its verification through experiments should provide good support for the correctness of our theory.

To test this lemma, we first initialized an array of $\hat{t}$ counters to zeros. Next, we fixed a one-way function by randomly choosing a plaintext and constructed a DP table with the fixed function. Then, a random password (= zero-padded encryption key) was generated and the password hash (= masked ciphertext) corresponding to that password was computed. The online chain starting from the password hash was computed until a DP was found or the $\hat{t}$-th iteration was reached. If the online chain terminated at a DP and it was found to reside within the DP table, the counter corresponding to the current online iteration count was incremented. The online chain generation was repeated multiple times with the same table, but with newly generated random keys. Note that, since we are not using perfect tables, it is possible for the online chain to collide simultaneously with more than one entry of the DP table. Care was taken to increment the counter corresponding to the current iteration count as many times as the number of collisions found. The whole process described after the counter initialization step was repeated multiple times, with each repetition using a newly generated one-way function and a DP table.

The test results for four different parameter sets are presented in Figure 7. Each of these experiments was done with 2000 tables and 5000 random online chains per table. In each of the four boxes, the barely visible thin dashed line represent our theory as given by Lemma 43. There are $\hat{t}$-many tiny dots in each box and these represent our experiment results. The height of the i-th dot, counting from the left, is the value of the i-th iteration counter at the end of the experiment divided by 2000×5000, the total number of chains that were utilized. All the experiment results match our theory very well.

C.3 Ending point truncation

Finally, we test the validity of our arguments concerning the ending point truncation method for reducing storage. The straightforward approach would be to simply test Lemma 23, Lemma 29, and Lemma 35 that

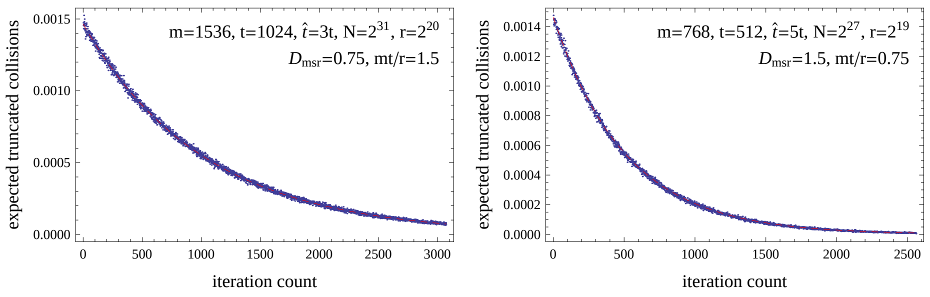


Fig. 8 Expected number of collisions, induced by ending point truncation, at each iteration of the DP tradeoff (dots: experiment; dashed line: theory)

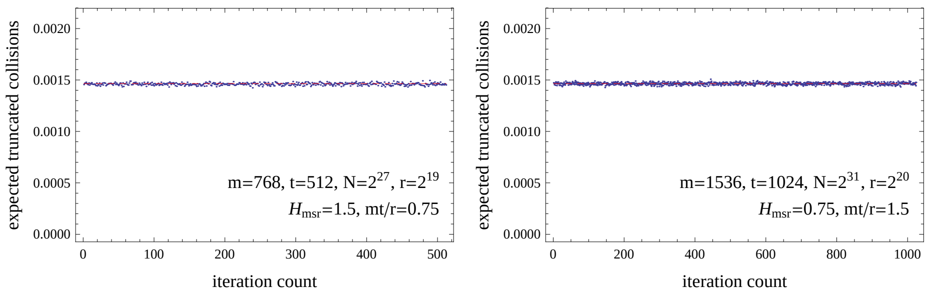


Fig. 9 Expected number of collisions, induced by ending point truncation, at each iteration of the Hellman tradeoff (dots: experiment; dashed line: theory)

present the cost of truncation related alarms, but we decided to work with the probability of alarms related to truncations, so as to expose more of our argument details to the tests.

Lemma 44 *Consider the DP tradeoff that uses ending point truncation of $\frac{1}{r}$ truncated match probability. At the i-th iteration of the online processing of a single DP table, the number of* pseudo-collisions *that are due to the ending point truncations, i.e., those that are not associated with any true chain collisions, is expected to be $\frac{m}{r}\exp(-\frac{i}{t})$.*

The corresponding value for the Hellman tradeoff is $\frac{m}{r}$, and that for the rainbow tradeoff is also $\frac{m}{r}$, if one decides to fully process a single rainbow table without terminating, even when the correct password is found.

Proof The proof to Lemma 23 shows that the claimed expected value for the DP tradeoff case can be computed as

$$\sum_{j=1}^{\hat{t}} \frac{\frac{m}{t}}{1-e^{-\hat{t}/t}} \exp\left(-\frac{j}{t}\right)\cdot\exp\left(-\frac{i}{t}\right)\frac{1}{r} \approx \frac{m}{1-e^{-\hat{t}/t}}\int_0^{\hat{t}/t}\exp(-v)\,dv\,\exp\left(-\frac{i}{t}\right)\frac{1}{r},$$

which simplifies to what is claimed. The statement for the Hellman tradeoff case follows immediately from the proof of Lemma 29, and the rainbow tradeoff case can be inferred from the proof of Lemma 35. □

The three claims given by this lemma are at the core of our arguments concerning the ending point truncation method, and experimental verification of these statements should provide some confidence to the validity of our arguments given in the main text.

As in the previous section, we generated random tradeoff tables and tested with random online chains for the occurrence of alarms induced from truncations. We stored the full ending points, together with the truncated ending points, in the pre-computation table. This was used to distinguish between alarms that were caused by ending point truncations and those that arose from true chain collisions.

The test results are given in Figure 8, Figure 9, and Figure 10. As before, the thin dashed lines are the graphs claimed in Lemma 44 and the numerous tiny dots represent experiment data. All the test results are in good agreement with the theory.

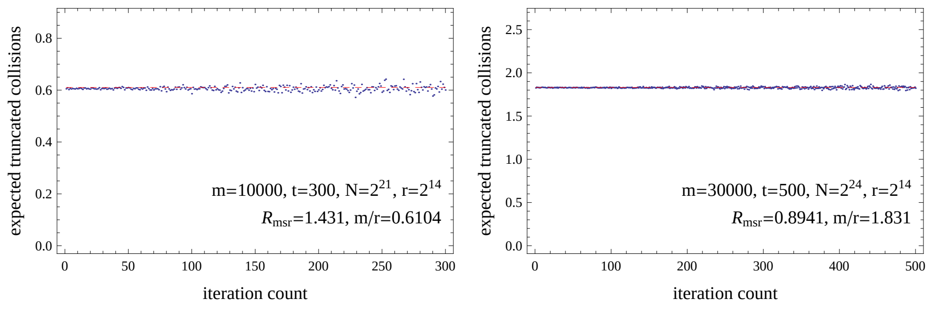


Fig. 10 Expected number of collisions, induced by ending point truncation, at each iteration of the rainbow tradeoff (dots: experiment; dashed line: theory)

Each of the two diagrams for the DP tradeoff was obtained by averaging over 2000 tables and 5000 chains per table. For the Hellman tradeoff we generated 2000 tables and 5000 inversion targets per table. The online chain was computed to the full length t for each inversion target and truncated match with the table elements was searched for after each one-way function iteration. In the rainbow tradeoff case, each diagram is the result of 100 tables with 5000 inversion targets per table. Recall that the k-th iteration for the rainbow tradeoff refers to a process that consists of $(k-1)$ invocations of the one-way function and one table lookup. Full t iterations were tried for each inversion target and hence each inversion target generated t searches to the table for truncated matches.