The Cost of False Alarms in Hellman and Rainbow Tradeoffs

Jin Hong

Department of Mathematical Sciences
Seoul National University, Seoul 151-747, Korea
jinhong@snu.ac.kr

Abstract. With any cryptanalytic time memory tradeoff algorithm, false alarms pose a major obstacle in accurate assessment of its online time complexity. We study the size of pre-image set under function iterations and use the obtained theory to analyze the cost incurred by false alarms. We are able to present the expected online time complexities for the Hellman tradeoff and the rainbow table method in a manner that takes false alarms into account. The effect of checkpoints in reducing the cost of false alarms is also quantified.

1 Introduction

Cryptanalytic time memory tradeoffs were first introduced by Hellman [11] as an attack on blockciphers. Since then, much progress has been made, with [1, 3–8, 10, 12, 13, 16] being a very partial list of works contributing to the theoretic aspects of the tradeoff algorithms. There are also numerous works on their applications and implementations. Today, the tradeoff algorithms are understood as techniques for quickly inverting generic one-way functions, and since much of cryptanalysis can be expressed as the process of inverting an appropriate one-way function, time memory tradeoff technique is a valuable tool for cryptography.

Any tradeoff algorithm works in two stages. During the pre-computation phase, the one-way function is iteratively computed and a digest of the computation is stored in tables of total size smaller than the complete dictionary. When a specific image point to be inverted is given, the online phase algorithm is supposed to return a pre-image of the target in time shorter than exhaustive search.

An analysis of the time and storage complexities is provided with any published tradeoff algorithm. For example, the tradeoff curve $TM^2 = N^2$ gives the online complexity of the original Hellman tradeoff, where N is the size of space on which the one-way function is defined. That is, for any T and M satisfying the tradeoff curve, the Hellman tradeoff algorithm, when run with appropriate parameters, enables one to finds a pre-image of the inversion target in time T, utilizing a pre-computed table of size M.

Tradeoff curves of various tradeoff algorithms are typically true only up to a small constant factor. This is largely due to the effects of what are called false alarms, which make it difficult to give an exact expression for the time complexity T. Most previous analyses of T concentrated on a certain main iterative process of the online phase and the added cost of dealing with false alarms were either neglected or roughly argued as being relatively small. For example, the original work by Hellman gave a heuristic argument that false alarms can increase the online computation by 50% at the most. On the other hand, the paper presenting the rainbow table method [16], which is the tradeoff method most widely known to the public, cites experiment results of observing 75% of cryptanalytic effort spent on false alarms. There, concerning false alarms, only a heuristic comparison to the distinguished point tradeoff method is given. Thus, it is apparent that the extra cost incurred by false alarms is not well understood, and this contributes a certain degree of uncertainty to the tradeoff curve.

The second source of obscurity in understanding the online complexity of a tradeoff algorithm from its tradeoff curve concerns the size of the pre-computed table. The value M in the tradeoff curve refers to the number of entries in the pre-computed table, while in any implementation of a tradeoff algorithm, the size of real world storage involved, rather than the number of entries, is of significance.

Table entry count does not translate directly to storage size, since the optimal storage size per entry is not made clear in any of the tradeoff algorithm analyses. Depending on the tradeoff algorithm, different storage techniques are applicable and some algorithms allow smaller entries to be used than others. Some of these storage reduction techniques, such as the use of essentially short inputs, are rather well understood, but there are less studied techniques that only the experienced are using. For example, each table entry may simply be truncated to a certain degree with only a small effect on the overall performance.

Thus we have two obstacles to understanding the exact online complexity of tradeoff algorithms. Furthermore, the two issues we have mentioned are not independent of each other. Some techniques for reducing storage, such as table entry truncation, results in slightly increased online time, and the checkpoint technique [1] designed to reduce the effects of false alarms requires extra storage.

The lack of understanding of the online complexities makes comparison between tradeoff algorithms difficult.[1] For example, the tradeoff curve for the rainbow table method [16] is $TM^2 = \frac{1}{2}N^2$ and this was argued as evidence of superiority over the Hellman tradeoff by a factor of two in time complexity. But the later work [4] asserted otherwise, mentioning difference in applicable storage reduction techniques. So, even though the behaviors of various tradeoff algorithms are known in terms of rough tradeoff curves, as the performance of these tradeoff algorithms differ mostly by a small factor, a fair comparison between algorithm performances is not a straightforward task.

[1] Contrary to what was said above, this author has experienced through reviews received on previous papers concerning the tradeoff technique, that some cryptographers believe our understanding of the tradeoff algorithms to be rather complete. Many clear-cut and strong statements on which tradeoff algorithm is better were received, but these did not point in the same direction.

As a first step in overcoming this difficulty, in this work, we give a more accurate assessment of the online time complexity, taking the effects of false alarms into account. The original Hellman tradeoff and the rainbow table method are considered. To the best of our knowledge, the only previous attempt at a rigorous analysis of false alarms appears in [1]. The work concentrates on the very special case of maximal perfect rainbow tables, which is difficult to use in practice, due to its high pre-computation cost. The current work will provide a realistic view of the cost incurred by false alarms and the resulting total online time complexity.

The rest of the paper is organized as follows. We start with a very brief review of the main concepts surrounding time memory tradeoffs in the next section. In Section 3, we consider an image set corresponding to a random input set under function iterations and study its pre-image size. The results obtained here are used to analyze the cost of false alarms in Section 4. This is the main contribution of this paper. Then, in Section 5, we show how the knowledge of iterated pre-image size can be used to study the effects of checkpoints in reducing false alarm costs. The final section summarizes this paper and provides comments on possible future developments of this work.

2 Tradeoff algorithms

A tradeoff algorithm is a method for inverting a generic one-way function. One is given a one-way function $F : \mathcal{N} \to \mathcal{N}$, acting on a finite set $\mathcal{N}$, together with an image point $F(x) \in \mathcal{N}$, and is asked to find x. There are also situations where one is given an image point $y \in \mathrm{Im}(F)$ and asked to find an $x \in \mathcal{N}$ such that $F(x) = y$. The difference between the two problems, which has mostly been ignored in the literature, is whether the objective is to obtain *the* solution or *any* solution to the inversion problem. For example, when attacking a certain password login system, one may be happy with *any* pre-image of the target password hash, but if the same password is suspected of being used in multiple systems with different hashing functions, *the correct* pre-image from one of the systems would be much more useful. In this work, we focus on the first problem, as it appears more naturally, but the second problem can be treated very similarly.

We shall not explain the explicit tradeoff algorithms and ask readers to refer to the original papers on Hellman tradeoff [11] and the rainbow tables [16]. Here, to set our working grounds, we will quickly review and fix some of the basic terminologies. Notation as given in this section will be used throughout this paper.

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

$$\mathrm{SP}_i = X_{i,0} \xrightarrow{F} X_{i,1} \xrightarrow{F} \cdots \xrightarrow{F} X_{i,t} = \mathrm{EP}_i \quad (1 \le i \le m). \tag{1}$$

We omit the *reduction functions*, as we shall deal with a single *Hellman table*. The complete set of m chains, consisting of $t + 1$ *columns* and m *rows*, is a *Hellman matrix*. In most cases, the parameters t and m are chosen to satisfy the *matrix stopping rule* $mt^2 = |\mathcal{N}|$. Likewise, we have the notions of a *rainbow chain*

$$\mathrm{SP}_i = X_{i,0} \xrightarrow{F_1} X_{i,1} \xrightarrow{F_2} \cdots \xrightarrow{F_t} X_{i,t} = \mathrm{EP}_i \quad (1 \leq i \leq m), \tag{2}$$

a *rainbow table*, and a *rainbow matrix*. It is usual to take $mt = \alpha|\mathcal{N}|$ with α close to 1. A rainbow table is *perfect* if all its ending points are distinct. A *maximal* perfect rainbow table is one containing the maximal number of possible rows.

If the current end $F^{k+1}(x)$ of the *online chain*

$$F(x) \xrightarrow{F} F^2(x) \xrightarrow{F} \cdots \xrightarrow{F} F^{k+1}(x)$$

of length k matches an *ending point* EP_i, we have an *alarm*. If an alarm involving EP_i shows the property $F^{t-k-1}(\mathrm{SP}_i) \neq x$, it is said to be a *false alarm*. Notice that this condition checks whether or not the correct x has been found, and is a weaker condition than $F^{t-k}(\mathrm{SP}_i) \neq F(x)$. There is a corresponding notion of false alarms for rainbow tables. False alarms are caused by *merges* between the online chain and the *pre-computed chain*.

Since we now have the basic terminology ready, let us briefly return to the *the correct* versus *any* pre-image versions of the inversion problem, discussed at the start of this section. The former problem corresponds to looking for x in the first t columns of the pre-computation matrix, i.e., among all matrix entries excluding the ending points. The latter problem is solved if $y = F(x)$ is found among matrix entries excluding the starting points. This difference affects the overall success probability of the tradeoff algorithm and needs to be kept in mind for any rigorous analysis of tradeoff algorithms.

Later, we shall deal with checkpoints [1]. This is a technique for resolving false alarms without regenerating the pre-computed chain, applicable to both Hellman and rainbow methods. The idea is to choose a fixed, say c-th, column of the pre-computation matrix and supplement a 1-bit information about $X_{i,c}$, for each i, in the pre-computed table. When an alarm sounds during the online phase, the corresponding value is computed for the online chain. If the online chain and the pre-computed chain had merged somewhere between the checkpoint and the ending point, there is a possibility that a comparison of checkpoint information will filter out the false alarm.

Throughout this paper, the one-way function to be inverted will always be written as $F : \mathcal{N} \to \mathcal{N}$, where $\mathcal{N}$ is a set of size N, and any Hellman matrix, denoted by `HM`, or a rainbow matrix, denoted by `RM`, will always consist of $t + 1$ columns and m rows.

3 Pre-image under function iteration

In this section, we present information concerning the size of a pre-image set under function iterations. While this paper focuses on time memory tradeoffs, we expect the results of this section to find other applications.

3.1 Ratio of i-nodes under function iteration

Given a random mapping $F : \mathcal{N} \to \mathcal{N}$, let us denote k iterations of F by $F^k = F \circ \cdots \circ F$. It is well known [9, 15] that if m_0-many distinct random inputs are subject to F^t, the expected image size m_t can be computed through the recursion

$$\frac{m_{k+1}}{N} = 1 - \exp\big(-\frac{m_k}{N}\big). \tag{3}$$

For example, using $m_0 = N$, one can state that $1 - \frac{1}{e}$ of the total space will belong to the image space of F. We will use the closed form approximation

$$\frac{m_k}{N} \approx \frac{1}{N/m_0 + k/2}, \tag{4}$$

which can be found[2] in [2].

Recall that an i-node is an element of $\mathcal{N}$ with exactly i-many pre-images. For a random mapping $F : \mathcal{N} \to \mathcal{N}$, it is known [9] that the ratio of i-nodes among $\mathcal{N}$ is expected to be

$$p_i = \frac{1}{e} \cdot \frac{1}{i!}, \tag{5}$$

as long as the set size N is large compared to i. Notice that $p_0 = \frac{1}{e}$ is in good agreement with our previous figure of $1 - \frac{1}{e}$ for the total image ratio.

For each non-negative integers i and k, let us write

$$\mathcal{R}_{i,k}(F) = \big\{y \in \mathcal{N} \mid y \text{ is an } i\text{-node under } F^k\big\}, \tag{6}$$

$$\mathcal{D}_{i,k}(F) = \big\{x \in \mathcal{N} \mid F^k(x) \in \mathcal{R}_{i,k}(F)\big\} \tag{7}$$

to denote the set of i-nodes and pre-images of i-nodes, associated to the mapping F^k. The characters $\mathcal{D}$ and $\mathcal{R}$ reflect the role of $\mathcal{N}$ as domain and range, respectively. We shall denote the i-node ratio under F^k by

$$p_{i,k} = \frac{|\mathcal{R}_{i,k}(F)|}{N}. \tag{8}$$

The values of $p_{i,1}$ has already been stated as p_i in (5).

Remark 1. A more technically correct definition for the above would be to express this as the expectation over all functions $F : \mathcal{N} \to \mathcal{N}$, but we simply take F to be a random function and refrain from going into more complicated expressions. Likewise, many of our future statements presenting explicit values should be understood as expectation over all function $F : \mathcal{N} \to \mathcal{N}$, or equivalently, for a random function, rather than for any fixed function.

We now define the formal power series

$$\mathcal{P}_k(x) = \sum_{i=0}^{\infty} p_{i,k} x^i, \tag{9}$$

[2] The statement in the referenced paper contains typographic errors.

for each non-negative integer k. For now, it is not clear if this is convergent for a real number input x. To answer this, we start with the following lemma, which allows an inductive approach.

Lemma 1. *The formal power series concerning the various i-node ratios satisfy the recurrence relation*

$$\mathcal{P}_0(x) = x, \quad \mathcal{P}_{k+1}(x) = \operatorname{Exp}\big(\mathcal{P}_k(x) - 1\big),$$

where $\operatorname{Exp}(x)$ *is the formal power series* $\sum_{i=0}^{\infty} \frac{1}{i!} x^i$.

Proof. As F^0 is the identity map, we have

$$p_{i,0} = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

and we have $\mathcal{P}_0(x) = x$. One can also substitute $p_{i,1}$, as given by (5), to obtain $\mathcal{P}_1(x) = \operatorname{Exp}(x-1)$, showing our claim to be true for at least the initial k.

To understand the general situation, let us fixed a single j-node for F and consider its j-many pre-images. If the number of pre-images under F^k of these j-many nodes add up to i, then the original j-node for F is an i-node for F^{k+1}. In fact, any i-node for F^{k+1} is some j-node for F, so it should not be too hard to convince oneself that

$$p_{i,k+1} = \sum_{j=0}^{\infty} p_j \Big(\sum_{i_1 + \cdots + i_j = i} p_{i_1,k} p_{i_2,k} \cdots p_{i_j,k} \Big),$$

with appropriate interpretation taken for empty products and the $i = 0$ case.

Now, notice that the coefficient of x^i in $\mathcal{P}_k(x)^j$ may be expressed as

$$\sum_{i_1 + \cdots + i_j = i} p_{i_1,k} p_{i_2,k} \cdots p_{i_j,k},$$

so that

$$\begin{aligned} \mathcal{P}_{k+1}(x) &= \sum_i \Big\{ \sum_j p_j \Big(\sum_{i_1 + \cdots + i_j = i} p_{i_1,k} p_{i_2,k} \cdots p_{i_j,k} \Big) \Big\} x^i \\ &= \sum_j p_j \sum_i \Big(\sum_{i_1 + \cdots + i_j = i} p_{i_1,k} p_{i_2,k} \cdots p_{i_j,k} \Big) x^i \\ &= \sum \frac{1}{e} \frac{1}{j!} \cdot \mathcal{P}_k(x)^j \\ &= \operatorname{Exp}\big(\mathcal{P}_k(x) - 1\big). \end{aligned}$$

This completes the proof. □

Recall that, for any real number x, the formal power series $\operatorname{Exp}(x)$ converges to the real value $\exp(x)$. Hence the above lemma shows that the formal power series $\mathcal{P}_k(x)$ is convergent for any real number x and non-negative integer k. We may now view $\mathcal{P}_k(x)$ as a function defined on the set of all real numbers.

Lemma 2. *The formal power series $\mathcal{P}_k(x)$, seen as a function, can be approximated as*

$$\mathcal{P}_k(x) \approx 1 - \frac{2(1-x)}{2+k(1-x)},$$

for $x \leq 1$.

Proof. The $x = 1$ case is trivially valid by Lemma 1. Let us temporarily use the notation $\mathcal{Q}_k(x) = 1 - \mathcal{P}_k(x)$. Then the recurrence relation of Lemma 1 can be written as

$$\mathcal{Q}_0(x) = 1 - x, \quad \mathcal{Q}_{k+1}(x) = 1 - \exp\bigl(-\mathcal{Q}_k(x)\bigr),$$

and $x < 1$ implies $\mathcal{Q}_k(x) > 0$ for all k.

Comparing this recurrence relation with (3) shows that we can use the approximation

$$\mathcal{Q}_k(x) \approx \frac{1}{1/\mathcal{Q}_0(x) + k/2},$$

as given by (4). The approximation is accurate when $0 < \mathcal{Q}_k(x) < \varepsilon$ is not too big, which corresponds to $1 - \varepsilon < x < 1$. We arrive at our claim by resubstituting $\mathcal{Q}_0(x) = 1 - x$ and $\mathcal{Q}_k(x) = 1 - \mathcal{P}_k(x)$ back into this approximation and simplifying the result. □

Thus, we have obtained a closed form approximation for $\mathcal{P}_k(x)$, which contains information about $p_{i,k}$.

3.2 Equivalence under function iterations

Let us define two points $x, x' \in \mathcal{N}$ to be F^k-equivalent, if $F^k(x) = F^k(x')$, which is trivially an equivalence relation. As an example use of this notion, recalling the notation (7), one can say that any point of $\mathcal{D}_{i,k}(F)$ is F^k-equivalent to i-many points, including itself.

In this subsection, we will work out the number of points that are F^k-equivalent to a set of m randomly chosen points. The single point case is relatively easy.

Lemma 3. *A random point of $\mathcal{N}$ is expected to be F^k-equivalent to $k+1$ points.*

Proof. A point of $\mathcal{N}$ is F^k-equivalent to i-many points if and only if it belongs to $\mathcal{D}_{i,k}$. Since $|\mathcal{D}_{i,k}| = i \cdot |\mathcal{R}_{i,k}|$, a random point of $\mathcal{N}$ belongs to $\mathcal{D}_{i,k}$ with probability $i \cdot p_{i,k}$. Hence, a single random point is expected to be F^k-equivalent to

$$\sum_i i(i \cdot p_{i,k}) = \sum_i i(i-1)p_{i,k} + \sum_i ip_{i,k} = \mathcal{P}_k''(1) + \mathcal{P}_k'(1) \qquad (11)$$

points.

Now, Lemma 1 shows $\mathcal{P}_k(1) = 1$, for all k, and $\mathcal{P}_0'(x) = 1$. We can also obtain $\mathcal{P}_{k+1}'(x) = \mathcal{P}_k'(x)\exp\bigl(\mathcal{P}_k(x) - 1\bigr)$ from the same lemma, which leads to $\mathcal{P}_k'(1) = 1$, for all k. Differentiation of what we already have shows $\mathcal{P}_{k+1}''(x) =$

$\big(\mathcal{P}_k''(x)+\mathcal{P}_k'(x)^2\big)\exp\big(\mathcal{P}_k(x)-1\big)$, which implies $\mathcal{P}_{k+1}''(1)=\mathcal{P}_k''(1)+1$. Starting with $\mathcal{P}_0''(1)=0$, we inductively obtain $\mathcal{P}_k''(1)=k$. Thus, we have $\mathcal{P}_k''(1)+\mathcal{P}_k'(1)=k+1$ and our claim is proved.

The last equality can also be obtained by directly differentiating the approximation for $\mathcal{P}_k(x)$, given by Lemma 2. □

Let $x\in\mathcal{N}$ be random and consider $y=F^k(x)$. By substituting $m_0=N$ into (4), we can state that the ratio of the complete F^k-image space size over the input space size is $\frac{2}{k+2}$. It is tempting to say that, on average, image point y will have $(\frac{2}{k+2})^{-1}=\frac{k}{2}+1$ pre-images, which implies that x is F^k-equivalent to $\frac{k}{2}+1$ points. This argument disagrees with the $k+1$ given by Lemma 3. The apparent contradiction arises from the careless use of randomness. The random choice of x leads to non-uniform choice among the image points. Those image points with larger pre-image sets are more likely to be taken than other image points. In the above lemma, we computed the pre-image size of a random-input image. This is different from the pre-image of a random point taken from the image space.

We may conclude from the above lemma that, for small m, the number of points F^k-equivalent to m random points will be $m(k+1)$. But when m is large, this is no longer true, as some of the m points will be equivalent to each other.

Lemma 4. *Let $D\subset\mathcal{N}$ be a set of $\bar{m}$ randomly chosen points. The number of points F^k-equivalent to points of D is expected to be approximately*

$$\bar{m}\big(k+\frac{\bar{m}k^2}{4N}+1\big)\big(1+\frac{\bar{m}k}{2N}\big)^{-2}.$$

Proof. Let us consider the process of choosing points of the F^k-image space through random selection of points in its domain $\mathcal{N}$. As discussed earlier, since the probability of an image point being selected will depend on the number of its pre-images, this will not produce a random distribution of image points. On the other hand, note that random selection within $\mathcal{D}_{i,k}$ will lead to uniform distribution in the corresponding image space $\mathcal{R}_{i,k}$.

Since $\frac{|\mathcal{D}_{i,k}|}{N}=i\cdot p_i$, the number of elements belonging to $D\cap\mathcal{D}_{i,k}$ is expected to be $\bar{m}\cdot i\cdot p_{i,k}$. Modeling F as a random function, we can interpret

$$F^k(D\cap\mathcal{D}_{i,k})=F^k(D)\cap\mathcal{R}_{i,k}$$

as a set obtained by drawing $(\bar{m}\cdot i\cdot p_{i,k})$-many points from $\mathcal{R}_{i,k}$, uniformly at random, with replacements. Thus the size of this image set is expected to be approximately

$$\begin{aligned}&|\mathcal{R}_{i,k}|\Big\{1-\Big(1-\frac{1}{|\mathcal{R}_{i,k}|}\Big)^{\bar{m}\cdot i\cdot p_{i,k}}\Big\}\\ &\qquad\approx Np_{i,k}\Big\{1-\Big(1-\frac{1}{Np_{i,k}}\Big)^{\bar{m}\cdot i\cdot p_{i,k}}\Big\}\approx Np_{i,k}\Big\{1-\Big(1-\frac{\bar{m}}{N}\Big)^{i}\Big\}.\end{aligned}$$

To find the total number of points F^k-equivalent to D, noting that $\{D_{i,k}\}_{i=0}^{\infty}$ is a partition of the domain, one sums the number of pre-images of $F^k(D \cap \mathcal{D}_{i,k})$ to obtain

$$\sum_i i \cdot N p_{i,k} \Big\{ 1 - \Big(1 - \frac{\bar{m}}{N}\Big)^i \Big\} = N \Big\{ \mathcal{P}_k'(1) - \big(1 - \frac{\bar{m}}{N}\big) \mathcal{P}_k'\big(1 - \frac{\bar{m}}{N}\big) \Big\}.$$

Now, referring to Lemma 2, one can substitute $\mathcal{P}_k'(x) \approx \big(\frac{2}{2+k(1-x)}\big)^2$ into this equation and arrive at our claim after simplification. □

For applications to time memory tradeoffs, it is more useful to have the above equivalence count given in terms of the final image space size rather than the input size.

Proposition 1. *Let $D \subset \mathcal{N}$ be a set of randomly chosen points. If the number of distinct elements in $R = F^k(D)$ is m, then the pre-image of R under F^k is expected to be of size*

$$m\big(1 + k - \frac{mk}{2N} - \frac{mk^2}{4N}\big).$$

Proof. The relation (4), giving iterated image sizes, can be rewritten in the form

$$m_0 \approx \frac{N}{N/m_k - k/2}$$

and interpreted as follows: To obtain m_k distinct image points under F^k, one needs to use m_0-many distinct random inputs, where m_0 is as given by this equation.

Thus, we are looking for the number of points F^k-equivalent to points of D, with $|D| \approx \frac{N}{N/m-k/2}$. One can now substitute $\bar{m} = \frac{N}{N/m-k/2}$ into the equation of Lemma 4 and simplify to arrive at our claim. □

We emphasize that the randomness in this proposition refers to the selection of inputs rather than to the selection within the image space. A much shorter, but less informative, proof for this proposition is given in Appendix A.

4 Cost of false alarms

Let us use results of the previous section to quantify the effects of false alarms. We shall show that for the Hellman tradeoff and the perfect rainbow tables under typical parameters, 14.3% and 25.8%, of the total online time is spent in resolving false alarms, respectively. Non-perfect rainbow tables are also analyzed.

As with any analysis of tradeoff algorithms, we shall model F as a random function during our arguments.

4.1 Hellman tradeoff

Suppose, for the moment, that $m_0 t^2 = \alpha N$, with α not very large. Then, approximation (4) shows

$$m_t \approx \frac{N}{N/m_0 + t/2} = \frac{m_0}{1+\alpha/2t} = m_0\Big\{1 - \frac{\alpha}{2t} + \big(\frac{\alpha}{2t}\big)^2 - \cdots\Big\} \approx m_0, \qquad (12)$$

implying that the number of collisions among ending points is very small compared to the total number of rows. Thus the behavior of the tradeoff algorithm will depend very little on whether or not we replace the colliding ending points with new chains. Hence, in our further discussions of the Hellman tradeoff, we shall assume that the Hellman matrix was created with $m_0 = m$ distinct starting points and that the resulting $m_t = m$ ending points are distinct. This insures, in particular, that all entries within each column of the Hellman matrix are distinct.

Hellman tradeoff utilizes multiple tables with usually one of these containing the correct answer. Unless all the tables are processed in parallel, we will end up producing the full length online chain for most of the Hellman tables that we start search on. So we shall state our cost claims in terms of single table processing, rather than for the whole online phase. If needed, one can readily obtain the expected total cost from the success probability of a single table and the cost per table.

The following is evident from the Hellman tradeoff algorithm itself.

Proposition 2. *Disregarding the effects of false alarms, full online processing of a single Hellman table requires $t-1$ applications of F.*

For $0 \le k \le t$, the k-th column of a Hellman matrix $\mathtt{HM}$ will be denoted by $\mathtt{HM}_k = \{X_{i,k}\}_{i=1}^m$, where $X_{i,k}$ is as given by (1). We shall write $F^{-j}(\mathtt{HM}_t)$ for the set of pre-images under F^j of the ending points $\mathtt{HM}_t$.

Lemma 5. *Let $mt^2 = \alpha N$. For each $0 < k \le t$, the size of the pre-image set $F^{-k}(\mathtt{HM}_t)$ is approximately $m(k+1)$, when $\frac{\alpha}{t}$ is small.*

Proof. Under our conditions, we have

$$\begin{aligned} m(1+k-\frac{mk}{2N}-\frac{mk^2}{4N}) &\approx m(1+k-\frac{m}{4N}-\frac{mk}{2N}-\frac{mk^2}{4N}) \\ &= m(1+k)\Big\{1-\frac{m(1+k)}{4N}\Big\} \approx m(1+k). \end{aligned}$$

According to Proposition 1, our claim is true for at least the $k = t$ case. Dealing with the $k < t$ case is an exercise in the definition of a random function. Consider a set of m fixed ending points and let D be the set of random inputs used in creating the m ending points. Since $F^{t-k}(D)$ is a set of points producing m distinct points under F^k, it suffices to argue that $F^{t-k}(D)$ is a random set of elements from $\mathcal{N}$. But this randomness is a consequence of F being modeled as a random function. $\square$

It is now possible to compute the extra work incurred by false alarms.

Proposition 3. *Let $mt^2 = \alpha N$. During the full online processing of a single Hellman table, false alarms are expected to cause approximately $\frac{\alpha}{6}t$ extra applications of F.*

Proof. Let $y = F(x)$ be given as the target image point. For each $0 < k \leq t$, the k-th iteration of the online phase, i.e., the search of $F^{k-1}(y)$ among the ending points, causes a false alarm if and only if $x \in F^{-k}(\mathtt{HM}_t)$ but $x \notin \mathtt{HM}_{t-k}$. By Lemma 5 and the assumption on ending point distinctness, the probability of such an incident happening is $\frac{m(k+1)-m}{N}$.

Each false alarm causes $(t - k + 1)$ iterations of F to be executed before it can be dismissed. Thus, the expected number of F iterations spent on false alarm treatment throughout the full processing of a single Hellman table can be calculated to be

$$\sum_{0<k\leq t} (t-k+1)\frac{mk}{N} \;=\; \frac{t(t+1)(t+2)}{6} \cdot \frac{m}{N}.$$

It now suffices to apply the condition $mt^2 = \alpha N$. □

We have tested the above claim with small parameters satisfying $mt^2 = N$. When averaged over multiple tables and multiple inversion targets, the number of false alarms observed from the $(t-k)$-th column was linear in k and almost indistinguishable from our theory.

Note that the relative cost of false alarms is very sensitive to α. Under the matrix stopping rule $mt^2 = N$, approximately $\frac{1}{6}t$ iterations of F are spent on false alarms per Hellman table. In comparison, Proposition 2 states that the online chain generation requires approximately t iterations of F per Hellman table. So if the multiple Hellman tables are processed sequentially, false alarms will cause $\frac{1}{6} \approx 16.7\%$ increase in online time. In other words, $\frac{1}{7} \approx 14.3\%$ of the total online time is related to false alarms.

Notice that $(t-k+1)\frac{mk}{N}$, the term being summed in the above proof, viewed as a function of k, attains its maximum value near $k = \frac{t}{2}$. So, the effects of false alarms is at its greatest when the online chain has reached half its full length. Assuming that the correct answer is found after going through half of the precomputed data, this implies that we get the same $\frac{1}{6}$ factor increase in online time due to false alarms, even when all the Hellman tables are processed in parallel.

4.2 Perfect rainbow table

We shall concentrate on analyzing cost from a single perfect rainbow table. At the end of this subsection, we briefly explain how to work with multiple tables. The perfect rainbow table is not assumed to (but may) contain the maximal number of possible rows. While maximal perfect tables are easier to analyze, building them are very costly so that near maximal tables are used in practice.

Recall the notation (2). We shall write $F_\star$ when referring to multiple F_j without explicitly specifying the indices. To deal with rainbow tables, we redefine $F^k = F_{\star+k-1} \circ \cdots \circ F_{\star+1} \circ F_\star$ to be any k iterations of consecutive $F_\star$. The proof of Lemma 2 remains valid for this new F^k and leads to Proposition 1 being true for the new F^k.

Disregarding the effects of false alarms, the rainbow table method requires approximately $\frac{1}{2}t^2$ applications of $F_\star$ in fully processing a single table, but as there could be early exits from the online phase algorithm, which occurs when the correct answer to the inversion problem is found, we give a measure that reflects the realistic use of rainbow tables.

Proposition 4. *When $mt = \alpha N$, disregarding the effects of false alarms, online processing of a single perfect rainbow table is expected to require approximately*

$$\left\{1 - (1+\alpha)e^{-\alpha}\right\}\left(\frac{t}{\alpha}\right)^2$$

applications of $F_\star$ in creating the online chain.

Proof. The inversion target y is in the last column with probability $\frac{m}{N}$. Checking for this requires no computation. This fails with probability $1 - \frac{m}{N}$, and in the second iteration, a single application of F_{t-1} is required. At the third iteration, which will be reached with probability $(1 - \frac{m}{N})^2$, application of $F_{t-2} \circ F_{t-1}$, or two applications of $F_\star$, is required.

Thus, the expected number of $F_\star$ iterations can be written as

$$\sum_{0<k\leq t} (k-1)\left(1 - \frac{m}{N}\right)^{k-1}. \tag{13}$$

Simplification of this, using the approximation $(1 - \frac{m}{N})^t \approx \exp(-\frac{mt}{N}) = e^{-\alpha}$, results in

$$\left\{\left(1 - \frac{\alpha}{t}\right) - \left(1 - \frac{\alpha}{t} + \alpha\right)e^{-\alpha}\right\}\left(\frac{t}{\alpha}\right)^2.$$

Collection of the highest terms in t from this equation is the claimed count. $\square$

We can similarly state the extra work caused by false alarms. The k-th column of the rainbow matrix is denoted by $\mathtt{RM}_k$, and we shall write $F^{-j}(\mathtt{RM}_t)$ for the set of pre-images under F^j of the ending points.

Proposition 5. *When $mt = \alpha N$, the online processing of a single perfect rainbow table is expected to incur approximately*

$$\left\{(2 - \alpha^2)e^{-\alpha} - 2(1-\alpha)\right\}\left(\frac{t}{2\alpha}\right)^2$$

applications of $F_\star$ in dealing with false alarms.

Proof. As was noted in the proof of Proposition 4, the k-th ($0 < k \leq t$) iteration is executed with probability $(1 - \frac{m}{N})^{k-1}$. As in the proof of Proposition 3, the k-th iteration sounds a false alarm if and only if $x \in F^{-k}(\mathtt{RM}_t) \setminus \mathtt{RM}_{t-k}$. As in the proof of Lemma 5, we may view $\mathtt{RM}_{t-k}$ as an F^k-image of random inputs. Then, Proposition 1 can be applied to shows that the above event happens with probability $\frac{m}{N}(k - \frac{mk}{2N} - \frac{mk^2}{4N})$. Each verification of whether we are dealing with a false alarm requires $(t - k + 1)$ iterations of $F_\star$.

The expected number of extra $F_\star$ iterations required to deal with false alarms is thus

$$\sum_{0<k\leq t} (t-k+1) \cdot \frac{m}{N}\Big(k - \frac{mk}{2N} - \frac{mk^2}{4N}\Big) \cdot \Big(1 - \frac{m}{N}\Big)^{k-1}. \tag{14}$$

This simplifies to

$$\frac{1}{4\alpha^2}\left\{\begin{array}{c} t^2\big(-2(1-\alpha) + (2-\alpha^2)e^{-\alpha}\big) \\ + t\alpha\big((4-\alpha) - (4+2\alpha-\alpha^2)e^{-\alpha}\big) \\ + 2\alpha^2\big(-1 + (1+\alpha)e^{-\alpha}\big) \end{array}\right\},$$

when the approximation $(1-\frac{m}{N})^t \approx \exp(-\frac{mt}{N}) = e^{-\alpha}$ is used. The highest terms in t from this equation is what our claim states. $\square$

We have tested the above claim with small parameters, averaging over multiple tables and multiple inversion targets. The number of false alarms occurring at each column was almost indistinguishable from our theory. A graph of this is provided as Figure 2 in Section 5.2.

The proposition shows that the relative cost of false alarms is very sensitive to $\alpha = \frac{mt}{N}$. When the parameters satisfy $mt = N$, iterations stated by Proposition 4 is approximately $(1 - \frac{2}{e})t^2$ and the iterations due to false alarms given by Proposition 5 is approximately $\frac{1}{4e}t^2$. This translates to 34.8% extra $F_\star$ iterations due to false alarms. In other words, 25.8% of the online time is spent in resolving false alarms. More such information can be found in Table 4, given in Section 5.2.

It is easy to translate results of this section to the case when ℓ tables are processed in parallel. To obtain the expected $F_\star$ iteration counts per table, it suffices to replace the $(1 - \frac{m}{N})^{k-1}$ factors appearing in (13) and (14) with $(1 - \frac{m}{N})^{\ell(k-1)}$. Resulting formulas are no longer simple, but computable. If multiple tables are processed sequentially, counts for the ℓ-th table may be obtained by multiplying the factor $(1-\frac{m}{N})^{t(\ell-1)} \approx \exp(-\alpha(\ell-1))$ to the corresponding values given for a single table.

4.3 Non-perfect rainbow table

Let us now consider non-perfect rainbow tables, i.e, the case where *none* of the colliding pre-computed chains are removed. Partial results for this case, obtained in a manner different from the current work, appears in [14]. The case where one removes only some of the colliding chains is also conceivable, but such a case does not appear in the literature, and will not be dealt with here.

When a non-perfect rainbow table is in use, the expected number of $F_\star$ iterations for online chain creation is

$$\sum_{0<k\le t} (k-1) \prod_{j=1}^{k-1} \Big(1 - \frac{m_{t-j}}{N}\Big),$$

where each m_j is given by (3) with $m_0 = m$. When the approximation (4) is applied to this, we see cancelations within the product term and the above simplifies into

$$\begin{aligned}\sum_{0<k\le t} (k-1) \cdot \frac{2N + m(t-k)}{2N + m(t-1)} \cdot \frac{2N + m(t-k-1)}{2N + m(t-2)} \\ \approx \frac{1}{12} \cdot (\alpha^2 + 8\alpha + 24) \cdot \Big(\frac{t}{\alpha+2}\Big)^2,\end{aligned}$$

under the condition $mt = \alpha N$.

To deal with the cost of false alarms, note that if the current end of the online chain matches a common ending point of two pre-computed chains, both of them will have to be recreated for false alarm verification. Thus, for the purpose of analyzing false alarm costs, we can simply ignore collisions and treat colliding pre-computed chains as independent chains. So, using Lemma 3, the expected number of extra $F_\star$ iterations required to deal with false alarms can be written as

$$\begin{aligned}\sum_{0<k\le t} (t-k+1) \cdot \Big\{\frac{m}{N}\big(1+k\big) - \frac{m}{N}\Big\} \cdot \prod_{j=1}^{k-1} \Big(1 - \frac{m_{t-j}}{N}\Big) \\ \approx \frac{\alpha}{60} \cdot (3\alpha^2 + 20\alpha + 40) \cdot \Big(\frac{t}{\alpha+2}\Big)^2.\end{aligned}$$

where m_j are defined as before. The approximation is based on (4) and uses the notation $mt = \alpha N$.

Proposition 6. *When $mt = \alpha N$, online processing of a single non-perfect rainbow table is expected to require approximately*

$$\frac{1}{12} \cdot (\alpha^2 + 8\alpha + 24) \cdot \Big(\frac{t}{\alpha+2}\Big)^2$$

applications of $F_\star$, in creating the online chain, and

$$\frac{\alpha}{60} \cdot (3\alpha^2 + 20\alpha + 40) \cdot \Big(\frac{t}{\alpha+2}\Big)^2.$$

additional invocations of $F_\star$, in resolving false alarms.

Some example values for the above online cost expectations are given in Table 5 of Section 5.3.

5 Checkpoints

The previous section provides a complete solution to the online time complexity of the Hellman tradeoff and the rainbow table method. But since there is a trick called checkpoints, that can be applied to tradeoff algorithms to reduce the cost of false alarms, our analysis would not be complete without considering its effect.

We show how to use what we have learned about the pre-image tree structure of tradeoff tables as a tool for computing the most effective checkpoint positions. It turns out that, due to the difference in relative cost of false alarms, checkpoints are more useful on rainbow table method than on the Hellman tradeoffs.

5.1 Hellman tradeoff

Let us first deal with the Hellman tradeoff case. While we can give analysis for any number of checkpoints, the solutions do not simplify into one uniform formula, and the content of this section is best explained with examples. Furthermore, our analysis shows that using a large number of checkpoints is uncalled-for in most situations.

Let us say that $y = F(x)$ is given as the target image point. We assume that a 1-bit checkpoint has been placed at the $(t-c)$-th column, i.e., c iterations from the ending point. We shall write $F^{-j}(\mathtt{HM}_k)$ for the set of pre-images under F^j of the k-th column $\mathtt{HM}_k$ of a Hellman matrix.

It is clear that, for $k < c$, false alarms at the k-th iteration of the online phase cannot be filtered out with the checkpoint. Since the online chain available to the tradeoff operator starts from $y = F(x)$, rather than from x, the checkpoint information becomes useful starting from the $(c+1)$-th iteration.

Suppose that we observed an alarm in the $(c+d)$-th iteration, implying $x \in F^{-(c+d)}(\mathtt{HM}_t)$. Notice that $\mathtt{HM}_{t-(c+d)} \subset F^{-d}(\mathtt{HM}_{t-c}) \subset F^{-(c+d)}(\mathtt{HM}_t)$. If $x \in \mathtt{HM}_{t-(c+d)}$, the correct answer has been found. If $x \in F^{-d}(\mathtt{HM}_{t-c}) \setminus \mathtt{HM}_{t-(c+d)}$, we have a false alarm, but the online chain starting from x will merge with the precomputed Hellman chain before passing over the checkpoint and the checkpoint information is useless in resolving false alarms. Finally, if $x \in F^{-(c+d)}(\mathtt{HM}_t) \setminus F^{-d}(\mathtt{HM}_{t-c})$, the 1-bit checkpoint information will resolve false alarms with probability $1/2$. Thus, false alarms which are unresolved by the checkpoint occurs at the $(c+d)$-th iteration with probability

$$\frac{1}{N}\Big\{\big(|F^{-d}(\mathtt{HM}_{t-c})| - |\mathtt{HM}_{t-(c+d)}|\big) + \frac{1}{2}\big(|F^{-(c+d)}(\mathtt{HM}_t)| - |F^{-d}(\mathtt{HM}_{t-c})|\big)\Big\}.$$

It remains to fill in the various set sizes. Assumed distinctness of ending points imply $|\mathtt{HM}_{c+d}| = m$ and we know from Lemma 5 that $|F^{-(c+d)}(\mathtt{HM}_t)| = m(c+d+1)$. Recalling the discussion at the beginning of Section 4.1, we may take $\mathtt{HM}_{t-c}$ to have been obtained through random inputs to F^{t-c} and also believe that $|\mathtt{HM}_{t-c}| = m$. Hence, once again, Lemma 5 shows $|F^{-d}(\mathtt{HM}_{t-c})| = m(d+1)$ to be a good approximation. Finally, the probability of false alarm at the k-th iteration is

$$\mathrm{Prob}_{\mathrm{FA}}(k) = \begin{cases} \frac{mk}{N} & \text{for } 1 \le k \le c, \\ \frac{mk}{N} - \frac{mc}{2N} & \text{for } c < k \le t. \end{cases}$$

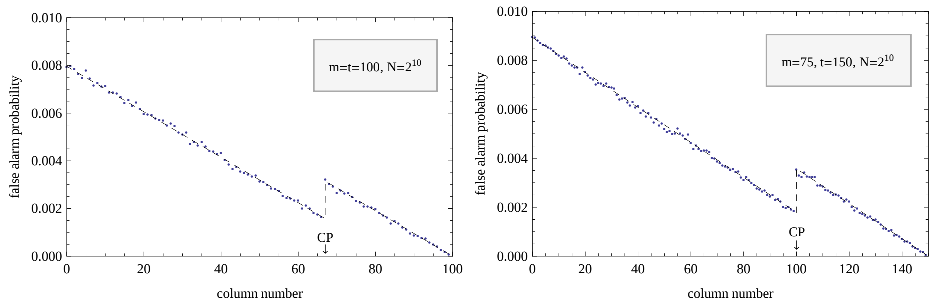

Fig. 1. Probability of false alarms for Hellman tradeoff with checkpoints (dashed line: theory, dots: experiment)

We have tested this, using small parameters. The results are given in Figure 1, where the $(t-k)$-th column corresponds to the k-th iteration. The one-way function used was the key to ciphertext mapping of AES-128 under a fixed random plaintext. To work with $N = 2^{20}$, the 20-bit input was zero-extended to a 128-bit key, and the 128-bit ciphertext was truncated to 20 bits. Each graph was obtained by averaging over 500 Hellman tables, each using a different input plaintext, and with 1000 random inversion targets per table. Ending point collisions were not removed during pre-computation and when an online chain matched the end of a pair of colliding chains, only one of them were tested for false alarms. The slightly irregular dots give the number of false alarms observed from each column divided by the total number of tests, i.e., 500000, and the dashed line, barely visible under the irregular dots, represent our theory.

Using the false alarm probability obtained, we can state that

$$\sum_{0<k\le t}(t-k+1)\frac{mk}{N} - \sum_{c<k\le t}(t-k+1)\frac{mc}{2N}$$

is the expected number of F iterations caused by false alarms. Let us summarize what has been discussed.

Example 1. By placing a single 1-bit checkpoint at the $(t-c)$-th column of a Hellman matrix, one can expect to remove

$$\sum_{c<k\le t}(t-k+1)\frac{mc}{2N} \;=\; \frac{m}{N}\left\{\frac{c(t-c)(t-c+1)}{4}\right\}$$

of the F applications caused by false alarms, per table. For large t, the maximum effect is obtained with $c \approx \frac{t}{3}$. When $mt^2 = \alpha N$, extra F iterations caused by false alarms, per table, can be reduced from approximately $\frac{\alpha}{6}t$ to $\frac{7}{54}\alpha t$, through the use of a single 1-bit check point at the optimal position.

One can easily extend the above results to more than one checkpoint. For example, when 1-bit checkpoints are placed at the $(t-c_2)$-th and $(t-c_1)$-th columns $(c_2 > c_1)$, the work induced by false alarms can be written as follows.

$$\begin{aligned}\sum_{0<k\le c_1} &(t-k+1)\big\{k\big\}\frac{m}{N} \\ &+ \sum_{c_1<k\le c_2} (t-k+1)\big\{(k-c_1)+\frac{c_1}{2}\big\}\frac{m}{N} \\ &+ \sum_{c_2<k\le t} (t-k+1)\big\{(k-c_2)+\frac{c_2-c_1}{2}+\frac{c_1}{4}\big\}\frac{m}{N}.\end{aligned}$$

Notice that alarms related to columns situated to the left of both checkpoints are filtered twice. We summarize this in a simplified form below.

Example 2. By placing 1-bit checkpoints at the $(t-c_2)$-th and $(t-c_1)$-th columns $(c_2 > c_1)$ of a Hellman matrix, one can expect to remove

$$\frac{1}{8}\frac{m}{N}\big\{(2c_1^3 - c_1c_2^2 + 2c_2^3) - (2c_1^2 - c_1c_2 + 2c_2^2)(2t+1) + (c_1+2c_2)(t^2+t)\big\}.$$

of the F iterations caused by false alarms, per table. For large t, this is approximately

$$\frac{1}{8}\frac{m}{N}\big\{(2c_1^3 - c_1c_2^2 + 2c_2^3) - (2c_1^2 - c_1c_2 + 2c_2^2)2t + (c_1+2c_2)t^2\big\},$$

and the maximum effect is obtained by using

$$c_1 \approx \frac{34-3\sqrt{46}}{53}t \approx 0.2576\,t \quad \text{and} \quad c_2 \approx \frac{29-\sqrt{46}}{53}t \approx 0.4192\,t.$$

Using these parameters, one can remove about $\frac{181+23\sqrt{46}}{5618}\alpha t \approx 0.05998\,\alpha t$ of the $\frac{\alpha}{6}t \approx 0.1667\,\alpha t$ function iterations related to false alarms.

# of CP	1-st CP	2-nd CP	3-rd CP	4-th CP
1 CP	0.33333			
2 CP	0.25760	0.41920		
3 CP	0.21166	0.33617	0.48067	
4 CP	0.18029	0.28272	0.39601	0.52748

Table 1. Optimal 1-bit checkpoint positions for Hellman tradeoff. (distance from ending point in units of t)

It is now clear how to approach any number of checkpoints. The computation will be more complicated, but clearly feasible for anyone that needs the information. We have done the computation and some optimal checkpoint positions are given in Table 1, where the values indicate distance from the ending points in units of t. Note that the optimal positions are independent of $\alpha = mt^2/N$.

Hellman	no CP	1 CP	2 CP	3 CP	4 CP
no CP		0.96825	0.94858	0.93505	0.92512
1 CP	62		0.97969	0.96571	0.95546
2 CP		97		0.98574	0.97527
3 CP			139		0.98938
4 CP				187	

Table 2. Relative online cost under different number of checkpoints.

In Table 2 we have listed the ratios for expected total number of F iterations when checkpoints are placed at the optimal positions. For example, the value 0.97969 in the table states that, compared to using a single checkpoint, using two checkpoints will result in 2.031% decrease in total online time.

Adding more checkpoints will require more storage space and may make the reduction in online time meaningless in view of the tradeoff $TM^2 \propto N^2$. The entries under the diagonal of Table 2 gives the minimum number of bits allocated to each Hellman table entry that would make the transition to higher number of checkpoints meaningful. For example, entry 97 means that, theoretically, unless each of the Hellman table entries took up at least 97 bits, the storage disadvantage of increasing the number of checkpoints from 1 to 2 outweighs the time advantage.

Of course, in practice, storage for checkpoints could essentially come for free due to properties concerning natural word size of the platform. Also, as discussed in the introduction, since a change in number of table entries affects the number of bits needed to store each entry, $TM^2 \propto N^2$ is not strictly true when M is taken to be the real storage size. So these numbers should be taken only as a rough guideline.

5.2 Perfect rainbow table

The general line of argument for analyzing the effects of checkpoints on perfect rainbow tables are exactly the same with the Hellman tradeoff case. We shall deal only with a single checkpoint on a single perfect rainbow table. Extension to multiple checkpoints is straightforward.

If a single checkpoint is placed at the $(t-c)$-th column, for each $k > c$, the probability of meeting a false alarm on the k-th iteration is

$$\frac{1}{N}\Big\{\big(|F^{-(k-c)}(\mathtt{RM}_{t-c})| - |\mathtt{RM}_{t-k}|\big) + \frac{1}{2}\big(|F^{-k}(\mathtt{RM}_t)| - |F^{-(k-c)}(\mathtt{RM}_{t-c})|\big)\Big\} \quad (15)$$

$$= \frac{1}{N}\Big\{\big(|F^{-k}(\mathtt{RM}_t)| - |\mathtt{RM}_{t-k}|\big) - \frac{1}{2}\big(|F^{-k}(\mathtt{RM}_t)| - |F^{-(k-c)}(\mathtt{RM}_{t-c})|\big)\Big\}. \quad (16)$$

According to Proposition 1, we have

$$|F^{-k}(\mathtt{RM}_t)| = m\big(1 + k - \frac{mk}{2N} - \frac{mk^2}{4N}\big), \quad (17)$$

and since perfectness implies $|\mathtt{RM}_{t-c}| = m$, just as was with $|\mathtt{RM}_t| = m$, we replace k in this equation with $k - c$ and state that

$$|F^{-(k-c)}(\mathtt{RM}_{t-c})| = m\big(1 + (k-c) - \frac{m(k-c)}{2N} - \frac{m(k-c)^2}{4N}\big) \quad (18)$$

is a reasonable approximation. There are subtle issues to be considered with this approximation and this is explained in Appendix B. We can now substitute (17) and (18) into (16). The second term within the braces corresponds to the false alarms that are filtered out, and after multiplying the work factor and the probability to reach the k-th iteration, we obtain

$$\frac{cm}{2N} \sum_{c<k\le t} (t-k+1)\Big(1 - \frac{m}{N}\Big)^{k-1}\Big(1 + \frac{m(c-2)}{4N} - \frac{m}{2N}k\Big), \quad (19)$$

as the number of F iterations that can be removed through a single 1-bit checkpoint at the $(t-c)$-th column.

α	0.25	0.5	1	1.5	1.8	1.9	1.95	2
optimal c/t	0.3193	0.3049	0.2724	0.2246	0.1827	0.1663	0.1577	0.1489

Table 3. Optimal single 1-bit checkpoint positions for perfect rainbow tables.

Unlike the Hellman tradeoff case, (19) does not simplify into a nice formula, but we can resort to numerical methods and still find the optimal checkpoint position. Let $mt = \alpha N$. It is easy to argue from (4) that, for any choice of parameters, we always have $\alpha < 2$. The optimal $(t-c)$-th column to place a single 1-bit checkpoint is given by Table 3, for some values of α spread over its range of interest.

In Table 4, we give the reduction in online iterations a checkpoint brings, when they are placed at the optimal position. The table lists expected numbers of total F iterations, iterations due to false alarms, and iterations reduced through checkpoints, per table. The count values are given in multiples of t^2. As the relative cost of false alarms is high on rainbow tables, the effects of checkpoints are better here compared to the Hellman tradeoff case.

α	0.25	0.5	1	1.5	1.8	1.9	1.95	2
total itr	0.4597	0.4222	0.3562	0.3014	0.2734	0.2648	0.2607	0.2566
FA itr	0.0357	0.0614	0.0920	0.1049	0.1076	0.1080	0.1081	0.1081
filtered FA itr	0.0077	0.0128	0.0176	0.0167	0.0136	0.0122	0.0114	0.0106
FA/total	7.77%	14.55%	25.82%	34.80%	39.37%	40.77%	41.45%	42.12%
filtered/total	1.67%	3.03%	4.94%	5.55%	4.98%	4.60%	4.37%	4.13%

Table 4. Iteration counts (unit: t^2) and effects of a single 1-bit checkpoint on a perfect rainbow table. (FA=false alarm, itr=iterations)

Remark 2. We give a word of caution for those interpreting data from Table 4. The iteration counts are given in multiples of t^2, so that, for example, the total iteration $0.3562t^2$ for $\alpha = 1$ being greater than $0.2566t^2$ for $\alpha = 2$ does not imply that the $\alpha = 1$ parameter takes longer. To the contrary, for the same m, the t for $\alpha = 2$ is twice as large as the t for $\alpha = 1$, and the net result is that $\alpha = 2$ takes much longer.

That said, this alone does not automatically imply superiority of the parameter $\alpha = 1$ over $\alpha = 2$, since the two brings about different (easily computable) success probabilities.

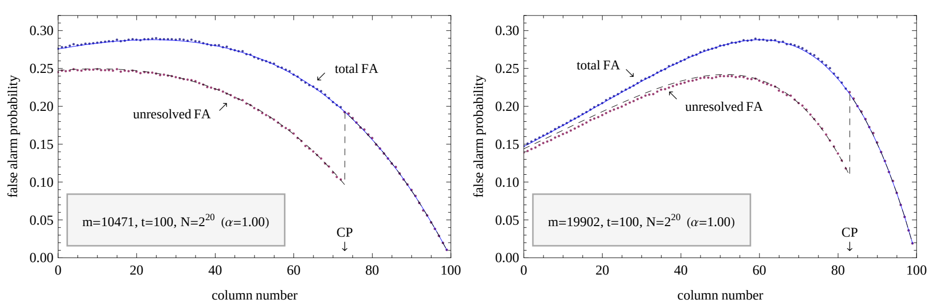

Fig. 2. Probability of reaching false alarms at each column for a perfect rainbow table. (lines: theory, dots: experiment)

We tested the theoretic result (19) with small parameters and this is shown in Figure 2. As with the Hellman tradeoff case, AES-128 was used to construct the one-way function. Both graphs were created with $N = 2^{20}$ and $t = 100$. Rather than fixing the number of collision free rows, we chose to fix m_0. The left hand side graph used $m_0 = 21020$ and the average of m_t observed over 30 tables was $m = 10471$, which translates to $\alpha = \frac{mt}{N} \approx 1.0$. The right hand side graph used $m_0 = 502418$. Average of m_t over 30 tables was $m = 19902$ and we have $\alpha \approx 1.9$. Each of the tables were subject to 5000 random inversion targets. The graphs give the number of false alarms observed from each column divided by the number of tests. The smooth solid and dashed lines are what our theory claims and the slightly irregular dots are the experiment results. A single checkpoint was used with each table at the optimal position and the lower graphs, in dashed lines, correspond to false alarms that remained unresolved after checkpoint information was utilized.

Our theory and test results for the number of all false alarms occurring at each column are indistinguishable. The observed number of false alarms that remained unresolved after checkpoint information use was slightly smaller than our theory at columns close to the starting points, but the cost $(t-k+1)$ of each false alarm is smaller there, so these discrepancies will have minimal effect on the total cost. The cause of this small discrepancy is explained in Appendix B.

5.3 Non-perfect rainbow table

Let us consider the non-perfect rainbow tables. After recalling the arguments of Section 4.3, one can treat the chains as totally independent and, unlike the perfect rainbow table case, the subtle issue considered in Appendix B no longer causes complications here.

The expected number of F iterations reduced through a single checkpoint at the $(t-c)$-th column can be computed as

$$\sum_{c<k\leq t} (t-k+1) \cdot \frac{m}{N}\frac{c}{2} \cdot \prod_{j=1}^{k-1} \Bigl(1 - \frac{m_{t-j}}{N}\Bigr).$$

Use of approximation (4) brings about cancelations within the last product term and the above becomes

$$\begin{aligned} &\frac{m}{N}\frac{c}{2} \sum_{c<k\leq t} (t-k+1) \cdot \frac{2N+m(t-k)}{2N+m(t-1)} \cdot \frac{2N+m(t-k-1)}{2N+m(t-2)} \\ &\qquad \approx \frac{\alpha}{24} \cdot \frac{c}{t} \cdot \bigl(1-\frac{c}{t}\bigr)^2 \cdot \Bigl\{3\alpha^2\bigl(1-\frac{c}{t}\bigr)^2 + 16\alpha\bigl(1-\frac{c}{t}\bigr) + 24\Bigr\} \cdot \Bigl(\frac{t}{\alpha+2}\Bigr)^2, \end{aligned}$$

under the notation $mt = \alpha N$.

α	0.25	0.5	1	2	5	10	100
optimal c/t	0.3218	0.3117	0.2952	0.2724	0.2410	0.2234	0.2026
total itr	0.4662	0.4443	0.4222	0.4208	0.5170	0.7431	5.2193
FA itr	0.0372	0.0677	0.1167	0.1917	0.3656	0.6250	5.1326
filtered FA itr	0.0082	0.0147	0.0250	0.0403	0.0755	0.1283	1.0512
FA/total	7.98%	15.23%	27.63%	45.54%	70.72%	84.12%	98.34%
filtered/total	1.75%	3.31%	5.91%	9.57%	14.61%	17.27%	20.14%

Table 5. Iteration counts (unit: t^2) and effects of a single 1-bit checkpoint at the optimal position on non-perfect rainbow tables. (FA=false alarm, itr=iterations)

One now has all the tools necessary to compute the optimal position to place a single 1-bit checkpoint. Table 5 lists the optimal $(t-c)$-th checkpoint column and various iteration counts for a non-perfect rainbow table. Note that the cost of false alarms quickly dominates the main online chain creation cost as α is increased.

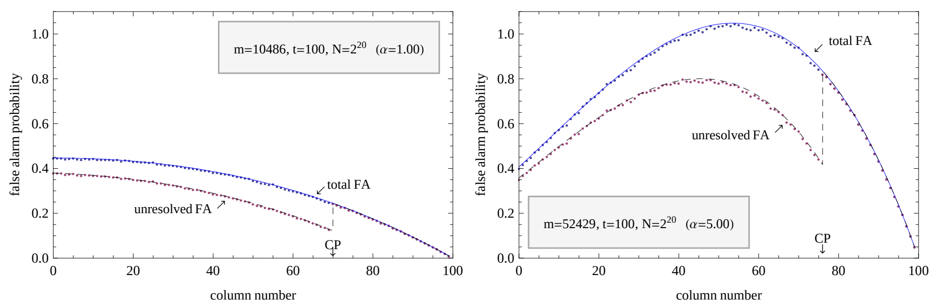

Fig. 3. Probability of reaching false alarms at each column for a non-perfect rainbow table. (lines: theory, dots: experiment)

We tested our claims with small parameters on one-way functions constructed from AES, and this is shown in Figure 3. Both graphs present data averaged over 30 tables, with 5000 random inversion targets per table. Readers may notice that the right hand side graph shows total false alarm probability greater than 1 at the center columns and consider this to be strange. This is an indication of multiple false alarm being expected in those columns, the effect of colliding ending points. Our theory and test results clearly agrees very well.

6 Conclusion

Previous analysis of the online time complexity for time memory tradeoff algorithms were usually based on the worst case operation of just the online chain creation process and the added cost of dealing with false alarms were either neglected or roughly argued as being relatively small.

In this work, we presented an accurate measure of the expected online time complexity, for the original Hellman and the rainbow table methods. By studying the size of pre-image sets under an iterated random function, the cost induced by false alarms were analyzed and taken into account. We have also analyzed the workings of the checkpoint method, computed their optimal positions, and quantified the resulting reduction in false alarm cost. The machinery developed for this analysis adds to our knowledge of the random function, and we hope it finds other applications.

For those familiar with the distinguished point method, we remark that more work needs to be done in order to give an analysis similar to the current work. There are many complications arising from the existence of distinguished points within the pre-image tree. In particular, chain length distribution and behavior of function iteration under domain restriction need to be considered.

While analyzing the effects of check points, we already experienced how time complexity may not be completely independent of storage size. The analysis of the second issue concerning online complexity that was discussed in the introduction, i.e., that of optimal storage size, will not be independent from this work. Storage reduction through aggressive ending point truncation brings about extra work in a probabilistic manner, just as with false alarms. Analysis of its effects and its optimal use will only be possible in conjunction with time complexity arguments.

Acknowledgement

The author thanks Daegun Ma for helpful comments.

References

1. G. Avoine, P. Junod, and P. Oechslin, Time-memory trade-offs: False alarm detection using checkpoints. *Indocrypt 2005,* LNCS 3797, pp.183–196, Springer-Verlag, 2005.
2. G. Avoine, P. Junod, and P. Oechslin, Time-memory trade-offs: False alarm detection using checkpoints (extended version). Technical Report LASEC-REPORT-2005-002, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASAC), Lausanne, Switzerland, September 2005.
3. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp.161–166, IEE, 1995.
4. E. Barkan, E. Biham, and A. Shamir, Rigorous bounds on cryptanalytic time/memory tradeoffs. *Crypto 2006,* LNCS 4117, pp.1–21, Springer-Verlag, 2006.
5. A. Biryukov, S. Mukhopadhyay, and P. Sarkar, Improved time-memory trade-offs with multiple data. *SAC 2005,* LNCS 3897, pp.110–127, Springer-Verlag, 2006.
6. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp.1–13, Springer-Verlag, 2000.
7. D. E. Denning, *Cryptography and Data Security* (p.100, Ron Rivest's distinguished points observation). Addison-Wesley, 1982.
8. A. Fiat and M. Naor, Rigorous time/space tradeoffs for invering functions. *SIAM J. on Computing,* **29**, no 3, pp.790–803, SIAM, 1999.
9. P. Flajolet and A. M. Odlyzko, Random mapping statistics. *Eurocrypt 1989,* LNCS 434, pp.329–354, Springer-Verlag, 1990.
10. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt 1997,* LNCS 1233, pp.239–255, Springer-Verlag, 1997.
11. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory,* **26** (1980), pp.401–406.
12. J. Hong, K. C. Jeong, E. Y. Kwon, I.-S. Lee, and D. Ma, Variants of the distinguished point method for cryptanalytic time memory trade-offs. *ISPEC 2008,* LNCS 4991, Springer-Verlag, pp.131–145, 2008.
13. I.-J. Kim and T. Matsumoto, Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Trans. Fundamentals,* **E82-A**, pp.123–129, 1999.

14. D. Ma, Studies on the cryptanalytic time memory trade-offs. *Ph.D. thesis,* Seoul National University, Aug., 2008.
15. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.
16. P. Oechslin, Making a faster cryptanalytic time-memory trade-off. *Crypto 2003,* LNCS 2729, pp.617–630, 2003.

A Another derivation for the equivalence set size

In this section, we present another proof[3] for Proposition 1, which gives the expected size of an F^k-equivalence set. The line of argument follows that of [1], where the special case corresponding to a maximal perfect rainbow table was considered.[4]

Given m and k, we set $m_k = m$ and recursively define $m_{k-1}, \dots, m_0$, through

$$m_{j-1} = -N \ln\big(1 - \frac{m_j}{N}\big).$$

This equation is equivalent to (3) and the m_j can be interpreted as follows. Suppose we begin with m_0 random starting points $D \subset \mathcal{N}$ and simultaneously create m_0 chains with iterated F-applications. At length i, the m_0 chains will have merged into m_i chains and will eventually end with $m_k = m$ distinct ending points $R = F^k(D)$ at length k.

Now, with these chains in place, construct a new chain of the same length, starting from a random point. That is, we are in the process of *defining* a random function and after the initial random definitions naturally produced chains ending at m distinct ending points, function definitions giving the last chain is being made.

With F modeled as a random function, the probability of the new chain not merging with the pre-constructed chain is $\prod_{i=0}^{k}(1 - \frac{m_i}{N})$. Notice that the random starting point of the new chain belongs to the pre-image $F^{-k}(R)$ if and only if the new chain merges with one of the pre-constructed chain. This shows that the size of pre-image set $F^{-k}(R)$ is

$$N\Big\{1 - \prod_{i=0}^{k}\big(1 - \frac{m_i}{N}\big)\Big\}.$$

The term inside the braces is the probability for a random point to be a pre-image point under F^k of the m ending points R and the product by N brings about the pre-image count.

[3] While this proof is provided under the belief that the general direction is correct, the author acknowledges that his understanding is not yet solid enough for himself to be comfortable with the proof.

[4] That the argument of [1] should be valid under more general circumstances was brought to the author's attention through an early version of [14].

It only remains to simplify this expression. Using the approximation (4), one can derive $\frac{m_{k-i}}{N} \approx \frac{1}{N/m - i/2}$. When this is substituted into the above, most of the product terms cancel out. What remains is

$$N\Big\{1 - \frac{(\frac{N}{m} - \frac{k}{2} - 1)}{(\frac{N}{m} - \frac{1}{2})}\frac{(\frac{N}{m} - \frac{k}{2} - \frac{1}{2})}{(\frac{N}{m})}\Big\} = N\Big\{1 - \Big(1 - \frac{\frac{k+1}{2}}{\frac{N}{m} - \frac{1}{2}}\Big)\Big(1 - \frac{\frac{k+1}{2}}{\frac{N}{m}}\Big)\Big\}$$
$$\approx N\Big\{1 - \Big(1 - \frac{m(k+1)}{2N}\Big)^2\Big\} \approx m\big(1 + k - \frac{mk}{2N} - \frac{mk^2}{4N}\big).$$

Thus, we have reobtained the explicit polynomial expression of Proposition 1.

Even though the proof given here requires less machinery than the proof in the main body of this paper, the longer proof shows more of the inner workings of the rainbow tables. For example, discussions of Appendix B would not have been possible without the insight obtained through the longer proof.

B Pre-image size of intermediate columns

The purpose of this section is to argue that (18) is a good approximation. In other words, we shall show that, for indices $0 \leq s < s+\delta \leq t$, the approximation

$$|F^{-\delta}(\mathtt{RM}_{s+\delta})| \approx m\big(1 + \delta - \frac{m\delta}{2N} - \frac{m\delta^2}{4N}\big), \tag{20}$$

is reasonable, when s is not too small.

B.1 Experimental evidence

Let us first provide some experiment data that supports our claim. The first set of graphs are given in Figure 4. We used parameters identical to those used in the

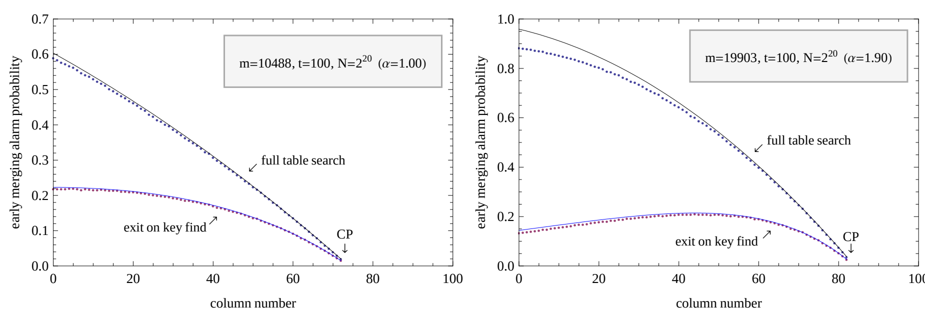

Fig. 4. Probability of encountering an alarm that corresponds to a chain merge before reaching the checkpoint. (lines: theory, dots: experiment)

testing of Section 5.2. In particular, the input counts were fixed to $m_0 = 21020$

and $m_0 = 502418$ for the two diagrams. As before, lines represent our theory and dots give the average obtained through tests. The graphs give the probability of reaching an (not necessarily false) alarm, having the property that the online chain merges with the pre-computed chain before passing over the checkpoint.

The upper graphs in the two boxed diagrams were obtained by going through the full t iterations of the online phase algorithm, regardless of whether the correct key was found. Hence these correspond to the actual pre-image sizes of $\mathtt{RM}_{t-c}$. The lower graphs give the probability that take into account the preliminary exits from the online phase algorithm on discovery of the correct key. Theoretically, this means that an extra $(1 - \frac{m}{N})^{t-k-1}$ term is multiplied at the k-th column.

The upper graphs show that the actual pre-image sizes is slightly smaller than what is given by (20), when we are near the starting points, i.e., when s is small. We can also see that the error is more prominent for the larger α. But, as is reflected in the lower graphs, the fact that the staring columns are not very likely to be reached during the online phase minimizes the effects of our error. Furthermore, the cost of resolving false alarms close to the starting columns are smaller than that of the ending columns. Hence our slightly large counts are good enough for the purpose of computing the expected cost of false alarms.

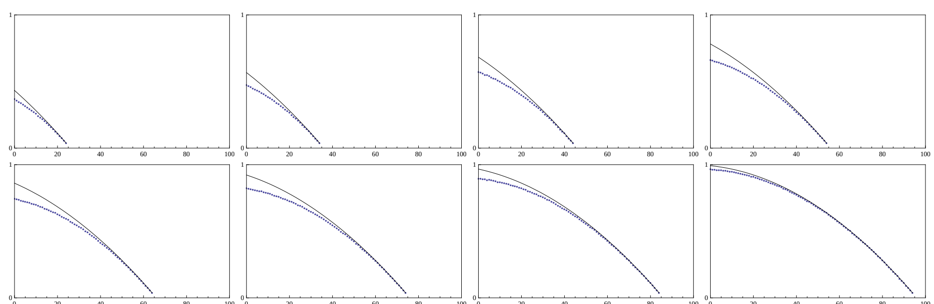

Fig. 5. Pre-image size of an intermediate column for $\alpha = 1.90$. (lines: theory, dots: experiment)

In Figure 5, we give some more experiment results for the $\alpha = 1.9$ case, which showed larger error than the $\alpha = 1.0$ case. Only the full table search graphs, corresponding to the pre-image sizes or the upper graphs, are provided, with the checkpoints at various positions. There are almost no error when the checkpoint is very close to the ending points. Also, regardless of the checkpoint position, the error is of noticeable size only when s is small.

B.2 Theoretic argument

Let us now give a logical argument showing (20) to be a reasonable approximation, when s is not too small. We start by explain why this is *not* implied by Proposition 1. Let D_0 be any random set that was used to create a rainbow table. What Proposition 1 gives is the pre-image size of $F^{s+\delta}(D_0)$, while we are looking for pre-image size of $\mathtt{RM}_{s+\delta}$ and because chains were discarded to remove ending point collisions, the column $\mathtt{RM}_{s+\delta}$ is only a partial subset of the random image $F^{s+\delta}(D_0)$. Hence Proposition 1 can not be used directly.

Now, define $D_k = F^k(D_0)$, for all k, so that $D_k \supset \mathtt{RM}_k = F^k(\mathtt{RM}_0)$. We are interested in the effects of selecting $\mathtt{RM}_{s+\delta}$ from $D_{s+\delta}$ on the pre-image sizes under F^δ. As we will be relating to i-nodes under multiple maps, let us refer to an i-node under F^k as an (F^k, i)-node. Since the size of a pre-image set, under a fixed F^δ, is completely determined by the size of the set to be inverted and the ratio of i-nodes (over all i) in that set, and since we already know

$$|F^{-\delta}(D_{s+\delta})| \approx m'\big(1 + \delta - \frac{m'\delta}{2N} - \frac{m'\delta^2}{4N}\big),$$

where $m' = |D_{s+\delta}|$, it suffices to show the following.

Claim. When s is not too small, for each i, the ratio of (F^δ, i)-nodes within $D_{s+\delta}$ is approximately equal to that within $\mathtt{RM}_{s+\delta}$.

Let us first review the collision removal method used during a rainbow table creation. The straightforward way to produce a perfect table is to create the rainbow chains sequentially, discarding any new chain that collides with an older chain, or replacing the older chain with the newer chain. Another approach is to create all chains from a random input set and then to randomly choose and keep a single chain from each group of colliding chains.[5] In either of the cases, within each F^t-equivalence class, the selection of $\mathtt{RM}_0$ from D_0 may be viewed as being random.

Since $\mathtt{RM}_k = F^k(\mathtt{RM}_0)$, the selection of $\mathtt{RM}_{s+\delta}$ within $D_{s+\delta}$ is dictated by the selection of $\mathtt{RM}_0$ within D_0. As the selection within each F^t-equivalence class of $\mathtt{RM}_0 \subset D_0$ is random, within each $F^{t-(s+\delta)}$-equivalence class, the probability of each point of $D_{s+\delta}$ being selected is governed by, for which i, it is an $(F^{s+\delta}, i)$-node.

Now, let us focus on a single point and consider its property of being an $(F^{s+\delta}, i)$-node and also an (F^δ, i')-node. When $s = 0$, we must have $i = i'$. Next, when s is small, there will be some correlation between i and i', i.e., if i is small, then i' is likely to be small, and when i is big, i' is also likely to be big. But this correlation quickly diminishes as we move to larger s. Furthermore, when F is modeled as a random function, the $F^{t-(s+\delta)}$-equivalence, which involves iterations to the right of $\mathtt{RM}_{s+\delta}$ is independent of any (F^δ, i)-node structure, which involves iterations to the left of $\mathtt{RM}_{s+\delta}$.

We have argued that the selection from within $D_{s+\delta}$ is governed by each point's $(F^{s+\delta}, \cdot)$-node and $F^{t-(s+\delta)}$-equivalence properties, and that neither of

[5] This approach will produce tables of slightly varying sizes, but addition or removal of small number of chains will have little effect on our analysis. Also, unless hash tables are used, this approach is much more efficient, as the sorting process automatically exposes collisions.

these properties are correlated to the $(F^{\delta}, \cdot)$-node property. Hence the selection of $\mathtt{RM}_{s+\delta}$ from $D_{s+\delta}$ is independent of the $(F^{\delta}, \cdot)$-node property, and the above Claim must hold true. In conclusion, the approximation (18) is valid, unless k is close to t.