Analysis of Multiple Checkpoints in Non-perfect and Perfect Rainbow Tradeoff Revisited*

Wenhao Wang[1,2] and Dongdai Lin[1]

[1] State Key Laboratory Of Information Security,
Institute of Information Engineering, CAS, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
{wangwenhao,ddlin}@iie.ac.cn

Abstract. Time memory tradeoff (TMTO) attack has proven to be an effective cryptanalysis method against block ciphers and stream ciphers. Since it was first proposed in 1980s, many new ideas have come out to reduce the false alarms during the online phase, among which rainbow table introduced by Oechslin and perfect table introduced by Borst et al. are notable landmarks. Avoine et al. introduced the checkpoints technique to detect false alarms using little additional memory without regenerating the pre-computed chain. In this paper, we revisit the analysis of multiple checkpoints in rainbow tradeoff. For non-perfect table we give a new sight to the computation of the expected decreasing number of chain regenerations at the k-th iteration. This helps to better understand the real nature of false alarms and leads us to the same results as the work of Jung Woo Kim et al. at Indocrypt 2012. For perfect rainbow tradeoff we give the first way to find optimal positions of multiple checkpoints. The results are better than previous work of Avoine et al., which only applies when the perfect table has the maximum number of chains. All the results are verified through meticulous experiments.

Keywords: time memory tradeoff, rainbow tradeoff, multiple checkpoints.

1 Introduction

Inverting one-way functions is one of the fundamental problems in cryptography. Much of cryptanalysis of block ciphers and stream ciphers can be expressed as the process of computation of pre-images or inversion of one-way functions. A cryptanalytic time-memory tradeoff (TMTO) is a technique to quickly invert generic one-way functions with the help of pre-computation. After it was first introduced by Hellman to perform an attack over DES [7] TMTO has been applied to many cryptosystems, for example against the GSM algorithm A5/1

* Supported by the National 973 Program of China under Grant 2011CB302400, the National Natural Science Foundation of China under Grants 10971246, 60970152, and 61173134, and the Strategic Priority Research Program of the Chinese Academy of Sciences under grant XDA06010701.

S. Qing et al. (Eds.): ICICS 2013, LNCS 8233, pp. 288–301, 2013.

[5], LILI-128 [16] and Windows LM Hash [15]. There are also ongoing research projects dealing with the implementations, such as the RainbowCrack Project [1] and the TMTO-based A5/1 Cracking Project [14] etc..

Two common ways to find a pre-image under a one-way function are exhaustive search and table lookup. In an exhaustive search method, one simply tries all possible keys in order to find the pre-image. With the table lookup method, one precomputes a table containing all the values of the key and then perform a search to find a pre-image. Hellman's TMTO attack achieves a middle ground between the exhaustive key search and the massive pre-computation of all possible ciphertexts for a given plaintext. During the pre-computation phase the attacker precomputes sufficiently many chains and only stores the starting points and ending points in a table. This concise table is used to find the pre-image in time shorter than an exhaustive search during an online phase.

A major drawback of Hellman's tradeoff attack is that it is possible to cause false alarms, when the online chain merges with pre-computed chains. False alarms significantly decrease the tradeoff efficiency and can increase more than 50% of the cryptanalysis time. After the inspiring work of Hellman, a lot of work has been done to reduce the cost of false alarms. Perfect table, suggested by Borst, Preneel, and Vandewalle in 1998, cleans the tables by discarding the merging and cycling pre-computed chains [6]. Rainbow table, introduced by Oechslin et al. in 2003, uses a different reduction function for each column of a table [15] and two different chains can merge only if they have the same key at the same position of the chain.

In 2005, Avoine, Junod and Oechslin [3] proposed using checkpoints to rule out false alarms. Additional information on some intermediate points of a chain are stored in the pre-computed tables, besides the starting points and ending points. During the online phase the attacker regenerates the pre-computed chain only when a match of both the ending point and the checkpoints is found. Using the technique, the cost of false alarms is reduced with a minute amount of memory. In [4] Avoine et al. presented an analysis of the effects of checkpoints in perfect rainbow tables when the table contains the MAXIMUM number of chains. And they did not clearly figure out the way to obtain optimal positions of multiple checkpoints in these tables. In 2010, Jin Hong et al. established a theoretical framework of analyzing false alarms [8] and gave a fair comparison of existing tradeoff algorithms [10]. Related works include the analysis of parallel distinguished point tradeoff [9], non-perfect table fuzzy rainbow tradeoff [11], perfect rainbow tradeoff [13] etc.. Analysis of one checkpoint for a single non-perfect rainbow table was done also in [8]. Analysis of multiple checkpoints for a non-perfect rainbow table was performed in [12].

In this paper we revisit the analysis of multiple checkpoints in rainbow tradeoff. For non-perfect rainbow tradeoff, when computing the expected decreasing number of false alarms at the k-th iteration, [12] used the approximation of $z_0 \approx m(1+k)$ and $z_u \approx m(1+k-c_u)$. They applied the same reasoning approach as [8] to analyze false alarm costs, by simply ignoring collisions and treating the pre-computed chains as independent chains. However this might not

be obvious[1]. We adopt a natural approach to compute the expected chains to be regenerated, with random selection of points at the k-th iteration. This leads us to identical results but explains the real nature of false alarms. For perfect rainbow tradeoff, considering that in most cases the perfect table has not the maximum number of chains due to the limit of pre-computation, we give the first way to find optimal positions of multiple checkpoints, even when the table is not maximum. All our results are verified through meticulous experiments.

The rest of the paper is organized as follows. We first introduce the theoretical framework of pre-image under function iteration and fix some notations in Section 2. In Section 3, we present our work on checkpoints for non-perfect rainbow tradeoff. In Section 4, we present our work on checkpoints for perfect rainbow tradeoff. In Section 5, we conclude the paper.

2 Theoretical Background

In this section we give some definitions that are used in the remainder of the paper.

2.1 Time Memory Tradeoff Attack

Let $F : \mathcal{N} \to \mathcal{N}$ be the one-way function to be inverted. In an off-line phase, we build m Hellman chains of length t with the form demonstrated in Fig. 1. A nice property of a chain is that we do not need to store all the elements in it. By knowing the starting point, we can recalculate the successive elements in the chain. So we just store the pairs of starting points and ending points $\{(SP_j, EP_j)\}_{j=1}^{m}$ in one table. Suppose l tables are constructed. A different reduction function r_i is used in the i-th table, and we denote $r_i(F(x))$ by $F_i(x)$. In the online phase, the goal is to find the unknown key by making use of the pre-computed tables. The attacker is given $y_0 = F(x_0)$ and has to find x_0. To search for x_0 in the i-th table, recursively he applies F_i to $r_i(y_0)$ and check if some $Y_k = (F_i)^k(x_0)$ appears as an ending point in the table. Whenever a match $Y_k = EP_j$ is found, he regenerates the corresponding pre-computed chain by computing $x = X_{j,t-k+1} = (F_i)^{t-k}(SP_j)$. There is a large chance that $(F_i)(x) = Y_1$, i.e. $F(x) = F(x_0)$.

Rainbow table uses a different reduction function for each column of a table. A rainbow chain is of the form

$$SP_j = X_{j,1} \xrightarrow{F} Y_{j,1} \xrightarrow{r_1} X_{j,2} \xrightarrow{F} Y_{j,2} \xrightarrow{r_2} \cdots \xrightarrow{F} Y_{j,t} \xrightarrow{r_t} EP_j,$$

and two different chains can merge only if they have the same key at the same position of the chain.

Checkpoint is a technique for resolving false alarms (false alarms occur if an online chain merges with pre-computed chains) without regenerating the chain,

[1] Note that the table becomes a perfect table if we treat the pre-computed chains as independent. And a perfect table is meant to cause less false alarms than a non-perfect table.

$$\begin{array}{rcl}
SP_1 = & X_{1,1} \xrightarrow{F} Y_{1,1} \xrightarrow{r} X_{1,2} \xrightarrow{F} Y_{1,2} \xrightarrow{r} \cdots \xrightarrow{F} Y_{1,t} & \xrightarrow{r} EP_1 \\
\vdots & & \vdots \\
SP_j = & X_{j,1} \xrightarrow{F} Y_{j,1} \xrightarrow{r} X_{j,2} \xrightarrow{F} Y_{j,2} \xrightarrow{r} \cdots \xrightarrow{F} Y_{j,t} & \xrightarrow{r} EP_j \\
\vdots & & \vdots \\
SP_m = & X_{m,1} \xrightarrow{F} Y_{m,1} \xrightarrow{r} X_{m,2} \xrightarrow{F} Y_{m,2} \xrightarrow{r} \cdots \xrightarrow{F} Y_{m,t} & \xrightarrow{r} EP_m
\end{array}$$

Fig. 1. Structure of a Hellman table

applicable to both Hellman and rainbow tradeoffs. Besides the starting points and ending points, additional information on some intermediate points are also stored. During the online phase the attacker regenerates the pre-computed chain only when a match of both the ending point and the checkpoints is found. Suppose a 1-bit information b_j about the intermediate point $X_{j,t-c}$ is extracted by a function G, i.e. $b_j = G(X_{j,t-c})$. During the online phase at the k-th iteration if an alarm is encountered and $k \geq c$, then we have

$$Pr\{b_j = G(Y_{k-c}) | X_{j,t-c} \neq Y_{k-c}\} \approx \frac{1}{2}.$$

Hence the comparison of checkpoint information can be used to filter out false alarms without regenerating the pre-computed chain.

2.2 Pre-image under Function Iteration

In this subsection we present previous results concerning the size of a pre-image set under an iteration of functions. The framework was established in [8] to analyze the cost of false alarms in Hellman tradeoff and rainbow tradeoff.

We consider a random one-way function $F : \mathcal{N} \to \mathcal{N}$, where $\mathcal{N}$ is a set of size N. Note that in Hellman tradeoff and rainbow tradeoff, the one-way function F is followed by a reduction function. Considering F as a random function, the following results are applicable to both Hellman tradeoff and rainbow tradeoff. We denote the k-times iteration of F by $F^k = F \circ \cdots \circ F$. It is well known that if m_0 distinct random inputs are subject to F^k, the expected image size denoted by m_k can be approximated by

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

Definition 1. *An i-node with respect to a mapping is an element of the range space with exactly i-many pre-images. For each non-negative integers i and k, let*

$$\mathcal{R}_{k,i}(F) = \{y \in \mathcal{N} | y \text{ is an } i\text{-node under } F^k\},$$
$$\mathcal{D}_{k,i}(F) = \{x \in \mathcal{N} | F^k(x) \in \mathcal{R}_{k,i}(F)\}$$

denote the set of i-nodes and pre-images of i-nodes, associated to the mapping F^k. An element of $\mathcal{R}_{k,i}(F)$ will be referred to as an (F^k, i)-node and the probability of a random point from the range space $\mathcal{N}$ to be an (F^k, i)-node is

$$p_{k,i} = \frac{|\mathcal{R}_{k,i}|}{N}.$$

For each non-negative integer k, fix a notation

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

for the formal power series relating to (F_k, i)-node ratios.

A closed form approximation for function $\mathcal{P}_k(x)$ is given in [8].

Theorem 1.

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

The number of points that are F^k-equivalent to a set of points is formulated by the following theorem, given the image space size. The theorem is frequently used by our later analysis and will not be explicitly specified.

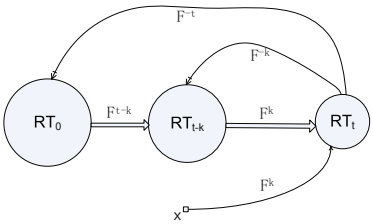
Theorem 2. *Let $D_0 \subset \mathcal{N}$ be a set of randomly chosen points. If the number of distinct elements in $D_t = F^t(D_0)$ is m_t, then the pre-image of D_t under F^k is expected to be of size*

$$m_t(1+k)(1 - \frac{m_t k}{4N}).$$

3 Checkpoints for Non-perfect Rainbow Tradeoff

In this section, we analyze the expected number of chains to be regenerated due to false alarms and the expected decreasing number of chain regenerations by checkpoints in a single non-perfect rainbow table. Note that if an online chain matches a common ending point of several pre-computed chains, then all these chains will have to be regenerated for false alarm verification. In [12] and [8], they compute the expected number of false alarms by simply ignoring the collisions in pre-computed chains and treating the pre-computed chains as independent chains. However the reasoning might not be obvious. We adopt a natural approach to compute the expected number of chains to be regenerated at the k-th iteration. This leads us to identical results but explains the real nature of false alarms.

We consider a non-perfect rainbow table constructed from $m_0 = m$ distinct starting points $RT_0 \subset \mathcal{N}$. Denote the chain length by t. For $1 \leq k \leq t$, denote the number of distinct elements in $RT_k = F^k(RT_0)$ by m_k, then we have $m_k \approx \frac{N}{N/m+k/2}$. We first study the expected number of pre-images of $F^k(x)$ under F^t that belongs to RT_0, with a random selection of $x \in \mathcal{N}$ at the k-th iteration.



Definition 2. *If we randomly choose a point $x \in \mathcal{N}$ and assume $F^k(x) \in RT_t$, then denote the number of pre-images of $F^k(x)$ under F^k that belongs to RT_{t-k} by $\mathcal{C}(k,k)$, and denote the number of pre-images of $F^k(x)$ under F^t that belongs to RT_0 by $\mathcal{C}(k,t)$.*

Proposition 1.

$$p_{k,i} = \begin{cases} \dfrac{k}{k+2} & i = 0 \\ \dfrac{4}{k(k+2)} \cdot (1 - \dfrac{2}{k+2})^i & i \geq 1 \end{cases}.$$

Proof. Recall from Theorem 1 that

$$\mathcal{P}_k(x) = \sum_{i=0}^{\infty} p_{k,i} x^i = 1 - \frac{2(1-x)}{2 + k(1-x)}.$$

For $i \geq 1$,

$$\mathcal{P}_k^{(i)}(x) = 4 \cdot i! \cdot k^{i-1} \cdot (2 + k(1-x))^{-(i+1)}.$$

According to the Maclaurin Series Expansion, we have

$$p_{k,i} = \frac{\mathcal{P}_k^{(i)}(0)}{i!} = \frac{4}{k(k+2)} \cdot (1 - \frac{2}{k+2})^i,$$

if $i \geq 1$. When $i = 0$, $p_{k,i} = \mathcal{P}_k(0) = \frac{k}{k+2}$. □

Proposition 2.

$$\mathcal{C}(k,k) = \frac{(\frac{m_{t-k} \cdot k}{N} + 2)^2}{\frac{m_{t-k} \cdot k}{N} + 4}, \quad \mathcal{C}(k,t) = \frac{(\frac{m_{t-k} \cdot k}{N} + 2)^2}{\frac{m_{t-k} \cdot k}{N} + 4} \cdot \frac{m}{m_{t-k}}.$$

Proof. Randomly choose a point $x \in \mathcal{N}$, denote by random variable A the number of pre-images of $F^k(x)$ under F^k. Since $p_{k,i}$ is the proportion of i-nodes in the domain space and every i-node has i pre-images under F^k, then the probability of a random x to produce an (F^k, i)-node is

$$Pr\{A = i\} = Pr\{F^k(x) \text{ is an } i\text{-node under } F^k\} = i \cdot p_{k,i}.$$

Denote by random variable B the number of pre-images of $F^k(x)$ under F^k that belongs to RT_{t-k}. Then for $j \leq i$, we have

$$Pr\{B = j|A = i\} = C_i^j(1 - \frac{m_{t-k}}{N})^{i-j}(\frac{m_{t-k}}{N})^j,$$
$$Pr\{B \neq 0|A = i\} = 1 - (1 - \frac{m_{t-k}}{N})^i.$$

By the law of total probability, for $j \geq 1$ we have

$$\begin{aligned} Pr\{B = j\} &= \sum_{i=j}^{N} Pr\{B = j|A = i\} \cdot Pr\{A = i\} \\ &= \sum_{i=j}^{N} C_i^j(1 - \frac{m_{t-k}}{N})^{i-j}(\frac{m_{t-k}}{N})^j \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i, \\ Pr\{B \neq 0\} &= \sum_{i=1}^{N} Pr\{B \neq 0|A = i\} \cdot Pr\{A = i\} \\ &= \sum_{i=1}^{N} [1 - (1 - \frac{m_{t-k}}{N})^i] \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i. \end{aligned}$$

Thus

$$\begin{aligned} \mathcal{C}(k,k) &= \sum_{j=1}^{m} j \cdot Pr\{B = j|B \neq 0\} \\ &= \sum_{j=1}^{m} \frac{j \cdot \sum_{i=j}^{N} C_i^j(1 - \frac{m_{t-k}}{N})^{i-j}(\frac{m_{t-k}}{N})^j \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i}{\sum_{i=1}^{N}[1 - (1 - \frac{m_{t-k}}{N})^i] \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i} \\ &= \frac{\sum_{j=1}^{m} j \cdot \sum_{i=j}^{N} C_i^j(1 - \frac{m_{t-k}}{N})^{i-j}(\frac{m_{t-k}}{N})^j \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i}{\sum_{i=1}^{N}[1 - (1 - \frac{m_{t-k}}{N})^i] \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i} \\ &= \frac{\sum_{i=1}^{N} \sum_{j=1}^{i} j \cdot C_i^j(1 - \frac{m_{t-k}}{N})^{i-j}(\frac{m_{t-k}}{N})^j \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i}{\sum_{i=1}^{N}[1 - (1 - \frac{m_{t-k}}{N})^i] \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i} \\ &= \frac{\sum_{i=1}^{N} i \cdot \frac{m_{t-k}}{N} \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i}{\sum_{i=1}^{N}[1 - (1 - \frac{m_{t-k}}{N})^i] \cdot \frac{4}{k(k+2)} \cdot i(1 - \frac{2}{k+2})^i} \\ &\approx \frac{\frac{m_{t-k}}{N} \cdot t^3 \int_0^\infty x^2 e^{-\frac{2t}{k+2}x}\, dx}{t^2 \int_0^\infty x(1 - e^{-\frac{m_{t-k}t}{N}x}) \cdot e^{-\frac{2t}{k+2}x}\, dx} \\ &\approx \frac{(\frac{m_{t-k} \cdot k}{N} + 2)^2}{\frac{m_{t-k} \cdot k}{N} + 4}. \end{aligned}$$

Given a random $x \in RT_{t-k}$, the expected number of pre-images of x under F^{t-k} that belongs to RT_0 is $\frac{m_0}{m_{t-k}}$, then we have

$$\mathcal{C}(k,t) = \mathcal{C}(k,k) \cdot \frac{m_0}{m_{t-k}} = \frac{(\frac{m_{t-k} \cdot k}{N} + 2)^2}{\frac{m_{t-k} \cdot k}{N} + 4} \cdot \frac{m}{m_{t-k}}.$$

□

Non-perfect Rainbow Tradeoff without Checkpoints. For a non-perfect table without checkpoints, at the k-th iteration during the online phase, the probability of an alarm is $Pr\{F^k(x) \in RT_t\} = \frac{1}{N}\{m_t(1+k)(1-\frac{m_t k}{4N})\}$. The expected number of chains to be regenerated for every alarm is $\mathcal{C}(k,t)$. Then the expected number of chains to be regenerated at the k-th iteration is

$$\mathcal{C}(k,t) \cdot \frac{1}{N}\{m_t(1+k)(1-\frac{m_t k}{4N})\} \approx \frac{m(1+k)}{N}. \quad (2)$$

Thus the expected number of chains to be regenerated due to false alarms at the k-th iteration is $\frac{1}{N}(m(1+k)-m)$. It is exactly what the authors claimed in [8].

Non-perfect Rainbow Tradeoff with n Checkpoints. Let $c_1, c_2, \cdots, c_n$ ($c_1 < c_2 < \cdots < c_n$) be the positions of n 1-bit checkpoints. That is the n checkpoints are located at the $(t-c_j)$-th columns for $j = 1, \cdots, n$. Let $c_0 = 0$ and $c_{n+1} = t$.

We compute the expected number of chains to be regenerated at the k-th iteration such that $c_j < k \leq c_{j+1}$ $(j = 1, \cdots, n)$. Given a random $x \in \mathcal{N}$, we have

$$Pr\{F^{k-c_j}(x) \in RT_{t-c_j}\} = \frac{m_{t-c_j}(1+k-c_j)}{N}[1 - \frac{m_{t-c_j}(k-c_j)}{4N}].$$

In such a case, an alarm always occurs and the expected number of chains to be regenerated is

$$Pr\{F^{k-c_j}(x) \in RT_{t-c_j}\} \cdot \mathcal{C}(k-c_j, t-c_j) = \frac{m(1+k-c_j)}{N}.$$

For $0 \leq u \leq j-1$,

$$Pr\{F^{k-c_u}(x) \in RT_{t-c_u}\} = \frac{m_{t-c_u}(1+k-c_u)}{N}[1 - \frac{m_{t-c_u}(k-c_u)}{4N}].$$

Then the expected number of pre-computed chains that merge with an online chain before the $(t-c_u)$-th column is

$$Pr\{F^{k-c_u}(x) \in RT_{t-c_u}\} \cdot \mathcal{C}(k-c_u, t-c_u) = \frac{m(1+k-c_u)}{N}.$$

Thus the expected number of pre-computed chains that merge with an online chain between the $(t-c_{u+1})$-th column and the $(t-c_u)$-th column is

$$\frac{m(1+k-c_u)}{N} - \frac{m(1+k-c_{u+1})}{N}.$$

In such a case, an alarm occurs with probability $1/2^{j-u}$. Therefore the expected number of chains to be regenerated at the k-th iteration such that $c_j < k \leq c_{j+1}$ $(j = 1, \cdots, n)$ is

$$\frac{m(1+k-c_j)}{N} + \sum_{u=0}^{j-1} \frac{1}{2^{j-u}} [\frac{m(1+k-c_u)}{N} - \frac{m(1+k-c_{u+1})}{N}].$$

Combined with Equation (2), we get the expected decreasing number of chains to be regenerated due to checkpoints. This simplifies to the same results as in [12].

Simulation Results. Our one-way function is built from a reduced version of MD5 hash function with $N = 2^{24}$. We built our pre-computed table from $m = 2^{16}$ different starting point and the chain length is $t = 300$. Table 1 shows that Proposition 2 agrees well with the experiment.

Table 1. Verification of Proposition 2

k	$\mathcal{C}(k,t)$	
	Theory	Experiment
50	1.6363	1.6333
100	1.6900	1.6916
150	1.7473	1.7230
200	1.8087	1.8199
250	1.8745	1.8661
300	1.9453	1.9507

4 Checkpoints for Perfect Rainbow Tradeoff

Perfect rainbow table is constructed by eliminating merged chains and thus reduces the cost of false alarms. Most rainbow tables available online are perfected before they are released to the public. It requires much more pre-computation for the generation of a perfect table, because chains with duplicate endpoints are removed.

We consider a perfect rainbow created with m_0 starting points and denote the chain length by t. Then we expect to collect $m = \frac{N}{N/m_0+t/2}$ non-merging chains. Let $r = \frac{m_0 t}{N}$ be the pre-computation coefficient. When a perfect table is created with $m_0 = N$ starting point, we have $m = \frac{N}{N/m_0+t/2} = \frac{2N}{t+2}$, and the table is referred to as a maximal perfect rainbow table. Let $\bar{r} = \frac{mt}{N}$, then $\bar{r} \leq \frac{2t}{t+2} \approx 2$. Table 2 shows that building a maximal perfect rainbow table (with $\bar{r}$ close to 2) is very costly and is seldom used in practice. An analysis of multiple checkpoints in maximal perfect tables was given in [4]. In this section, we give the first way

Table 2. Relations in a perfect rainbow table

r	$\bar{r}$	success rate	r	$\bar{r}$	success rate
1.0607	0.6931	50%	1.6911	0.9163	60%
3.0251	1.2040	70%	4.5178	1.3863	75%
8.2407	1.6094	80%	18.4363	1.8971	85%

to find optimal positions of multiple checkpoints even when the perfect table is not maximum.

Suppose n checkpoints are located at the $(t - c_j)$-th columns for $j = 1, \cdots, n$ $(c_1 < c_2 < \cdots < c_n)$. Let $c_0 = 0$ and $c_{n+1} = t$. In a perfect table for every false alarm only one chain need to be regenerated, so we only need to compute the probability of false alarms at the k-th iteration, for $c_j < k \leq c_{j+1}$ $(j = 1, \cdots, n)$. Given a random $x \in \mathcal{N}$, we have

$$Pr\{F^{k-c_j}(x) \in RT_{t-c_j}\} = \frac{m(1+k-c_j)}{N}[1 - \frac{m(k-c_j)}{4N}].$$

In such a case, an alarm always occurs. For $0 \leq u \leq j-1$,

$$Pr\{F^{k-c_u}(x) \in RT_{t-c_u}\} = \frac{m(1+k-c_u)}{N}[1 - \frac{m(k-c_u)}{4N}].$$

Thus the probability of a merge of the online chain with pre-computed chain between the $(t - c_{u+1})$-th column and the $(t - c_u)$-th column is

$$\frac{m(1+k-c_u)}{N}[1 - \frac{m(k-c_u)}{4N}] - \frac{m(1+k-c_{u+1})}{N}[1 - \frac{m(k-c_{u+1})}{4N}].$$

In such a case, an alarm occurs with probability $1/2^{j-u}$. Also the expected number of alarms at the k-th iteration without checkpoints is

$$Pr\{F^k(x) \in RT_t\} = \frac{m(1+k)}{N}(1 - \frac{mk}{4N}).$$

Therefore the expected decreasing number of false alarms at the k-th iteration such that $c_j < k \leq c_{j+1}$ $(j = 1, \cdots, n)$ is

$$\begin{aligned} D(k,j) &= \frac{m(1+k)}{N}(1 - \frac{mk}{4N}) - \left\{ \frac{m(1+k-c_j)}{N}[1 - \frac{m(k-c_j)}{4N}] \right. \\ &\left. + \sum_{u=0}^{j-1} \frac{1}{2^{j-u}} \left(\frac{m(1+k-c_u)}{N}[1 - \frac{m(k-c_u)}{4N}] - \frac{m(1+k-c_{u+1})}{N}[1 - \frac{m(k-c_{u+1})}{4N}] \right) \right\} \\ &= \frac{1}{N}\{(1 - \frac{1}{2^j}) \cdot m(1+k)(1 - \frac{mk}{4N}) - \sum_{u=0}^{j-1} \frac{1}{2^{j-u}} m(1+k-c_{u+1})(1 - \frac{m(k-c_{u+1})}{4N})\}. \end{aligned}$$

The k-th iteration of the online phase is executed with probability $(1 - \frac{m}{N})^k$ and every verification of a false alarm requires $(t - k + 1)$ iterations of F. This leads us to the following theorem.

Theorem 3. *The decreasing number of invocations of F due to n checkpoints $(c_1 < c_2 < \cdots < c_n)$ is*

$$S = \sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \cdot D(k,j) \cdot (1 - \frac{m}{N})^k \}. \tag{3}$$

Simplification. We rewrite Equation (3) to the sum of 4 parts.

$$S.1 = \sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \{ (1 - \frac{1}{2^j}) \cdot m(1+k)(1 - \frac{mk}{4N}) \cdot (1 - \frac{m}{N})^k \},$$

$$S.2 = -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \cdot \sum_{u=0}^{j-1} \frac{1}{2^{j-u}} m(1+k)(1 - \frac{mk}{4N}) \cdot (1 - \frac{m}{N})^k \},$$

$$S.3 = -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \{ m \cdot (\frac{m(2k+1)}{4N} - 1) \cdot \sum_{u=0}^{j-1} \frac{c_{u+1}}{2^{j-u}} \cdot (1 - \frac{m}{N})^k \} \},$$

$$S.4 = \sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \{ \frac{m^2}{4N} \cdot \sum_{u=0}^{j-1} \frac{c_{u+1}^2}{2^{j-u}} \cdot (1 - \frac{m}{N})^k \} \}.$$

Computation of S.2:

$$\begin{aligned} S.2 &= -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \cdot \sum_{u=0}^{j-1} \frac{1}{2^{j-u}} m(1+k)(1 - \frac{mk}{4N}) \cdot (1 - \frac{m}{N})^k \} \\ &= -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \cdot (\sum_{u=0}^{j-1} \frac{1}{2^{j-u}}) \cdot m(1+k)(1 - \frac{mk}{4N}) \cdot (1 - \frac{m}{N})^k \} \\ &= -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \cdot (1 - \frac{1}{2^j}) \cdot m(1+k)(1 - \frac{mk}{4N}) \cdot (1 - \frac{m}{N})^k \} \\ &= -S.1. \end{aligned}$$

Computation of S.3:

$$\begin{aligned} S.3 &= -\sum_{j=1}^{n} \{ \sum_{c_j < k \le c_{j+1}} (t-k+1) \frac{1}{N} \{ m \cdot (\frac{m(2k+1)}{4N} - 1) \cdot \sum_{u=0}^{j-1} \frac{c_{u+1}}{2^{j-u}} \cdot (1 - \frac{m}{N})^k \} \} \\ &\approx -\frac{mt^2}{N} \cdot \sum_{j=1}^{n} \{ [\sum_{c_j < k \le c_{j+1}} (1 - \frac{k}{t}) \cdot (\frac{mt}{2N} \cdot \frac{k}{t} - 1) e^{-\frac{mt}{N} \cdot \frac{k}{t}}] \cdot \frac{1}{t} \cdot \sum_{u=0}^{j-1} \frac{c_{u+1}}{2^{j-u}} \} \\ &\approx -\frac{mt^2}{N} \sum_{j=1}^{n} \{ \int_{c_j/t}^{c_{j+1}/t} (1-x) e^{-\frac{mtx}{N}} (\frac{mt}{2N} \cdot x - 1)\, dx \cdot \sum_{u=0}^{j-1} \frac{c_{u+1}}{2^{j-u}} \}. \end{aligned}$$

Computation of S.4:

$$S.4 = \sum_{j=1}^{n}\{\sum_{c_j<k\le c_{j+1}}(t-k+1)\frac{1}{N}\{\frac{m^2}{4N}\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}^2}{2^{j-u}}\cdot(1-\frac{m}{N})^k\}\}$$
$$= \frac{m^2t^2}{4N^2}\cdot\sum_{j=1}^{n}\{[\sum_{c_j<k\le c_{j+1}}(1-\frac{k}{t})e^{-\frac{mt}{N}\cdot\frac{k}{t}}]\cdot\frac{1}{t}\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}^2}{2^{j-u}}\}$$
$$= \frac{m^2t^2}{4N^2}\cdot\sum_{j=1}^{n}\{\int_{c_j/t}^{c_{j+1}/t}(1-x)e^{-\frac{mtx}{N}}\,dx\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}^2}{2^{j-u}}\}.$$

Thus Equation (3) simplifies to

$$S = -\frac{mt^2}{N}\sum_{j=1}^{n}\{\int_{c_j/t}^{c_{j+1}/t}(1-x)e^{-\frac{mtx}{N}}(\frac{mt}{2N}\cdot x-1)\,dx\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}}{2^{j-u}}\}$$
$$+\frac{m^2t^2}{4N^2}\cdot\sum_{j=1}^{n}\{\int_{c_j/t}^{c_{j+1}/t}(1-x)e^{-\frac{mtx}{N}}\,dx\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}^2}{2^{j-u}}\}.$$

Simulation Results. We experiment with $N = 2^{24}$, $m = 65536$, $t = 300$. Table 3 shows the experiment results with 3 checkpoints, located at the 177-th, 218-th and 251-th columns. We used Maple to obtain these optimal positions. The data are averaged over 10000 random inversion targets. The success probability (69.5%) is also close to the theoretical expectation (69.02%).

Table 3. Experiment for 3 checkpoints

	without checkpoint		with 3 checkpoints	
	Theory	Experiment	Theory	Experiment
#total operations	30255	30154	26687	26540
#operation for FA	8812	8833	5265	5219
#false alarms	69.4235	69.4529	42.8058	42.0807

5 Conclusion

Checkpoint is a useful technique to quickly rule out false alarms with a little additional memory. While the positions of checkpoints significantly affect the tradeoff efficiency, one of the key issue is to locate the optimal setting of multiple checkpoints. In this paper, we revisited the analysis of multiple checkpoints in rainbow tradeoff. For non-perfect table we gave a new sight to the computation of the expected decreasing number of false alarms at the k-th iteration. This helps to better understand the real nature of false alarms. For perfect rainbow table, we obtained the first full analysis applicable even if the perfect rainbow table is

not maximum. Considering in practice the table available are mostly perfect and not maximum, this part of our work is of value to the ongoing research projects, e.g. the RainbowCrack Project. Through experiment we saw a drastic decrease of cost due to false alarms, with only little additional memory.

References

1. Rainbowcrack project, `http://project-rainbowcrack.com/`
2. Avoine, G., Bourgeois, A., Carpent, X.: Discarding the endpoints makes the cryptanalytic time-memory trade-offs even faster
3. Avoine, G., Junod, P., Oechslin, P.: Time-memory trade-offs: False alarm detection using checkpoints. In: Maitra, S., Veni Madhavan, C.E., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 183–196. Springer, Heidelberg (2005)
4. Avoine, G., Junod, P., Oechslin, P.: Characterization and improvement of time-memory trade-off based on perfect tables. ACM Transactions on Information and System Security (TISSEC) 11(4), 17 (2008)
5. Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 1–18. Springer, Heidelberg (2001)
6. Borst, J., Preneel, B., Vandewalle, J.: On the time-memory tradeoff between exhaustive key search and table precomputation. In: Symposium on Information Theory in the Benelux, pp. 111–118. Citeseer (1998)
7. Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), 401–406 (1980)
8. Hong, J.: The cost of false alarms in hellman and rainbow tradeoffs. Designs, Codes and Cryptography 57(3), 293–327 (2010)
9. Hong, J., Lee, G.W., Ma, D.: Analysis of the parallel distinguished point tradeoff. In: Bernstein, D.J., Chatterjee, S. (eds.) INDOCRYPT 2011. LNCS, vol. 7107, pp. 161–180. Springer, Heidelberg (2011)
10. Hong, J., Moon, S.: A comparison of cryptanalytic tradeoff algorithms. Journal of Cryptology, 1–79 (2010)
11. Kim, B.-I., Hong, J.: Analysis of the non-perfect table fuzzy rainbow tradeoff. In: Boyd, C., Simpson, L. (eds.) ACISP. LNCS, vol. 7959, pp. 347–362. Springer, Heidelberg (2013)
12. Kim, J.W., Seo, J., Hong, J., Park, K., Kim, S.-R.: High-speed parallel implementations of the rainbow method in a heterogeneous system. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 303–316. Springer, Heidelberg (2012)
13. Lee, G.W., Hong, J.: A comparison of perfect table cryptanalytic tradeoff algorithms. Technical report, Cryptology ePrint Archive, Report 2012/540 (2012)
14. Nohl, K.: Attacking phone privacy. BlackHat 2010 Lecture Notes (2010)
15. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)
16. Saarinen, M.-J.O.: A time-memory tradeoff attack against LILI-128. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 231–236. Springer, Heidelberg (2002)

A When Multiple Tables Are Used

It is easy to translate the results to the case when l tables are processed in parallel. We just give the results and omit the proof.

Corollary 1. *For non-perfect rainbow tables with the parameters m, t, and l, where l is number of tables. Given n checkpoints $c_1 < c_2 < \cdots < c_n$, the expected number of f invocations that can be removed through checkpoints is*

$$l\sum_{j=1}^{n}\left\{\sum_{c_j<k\le c_{j+1}}(t-k+1)\cdot\frac{m}{N}\sum_{u=0}^{j-1}\left(\frac{c_{u+1}}{2^{j-u}}\right)\cdot\prod(1-\frac{m_{t-i}}{N})^{l}\right\}.$$

Corollary 2. *For perfect rainbow tables with the parameters m, t, and l, where l is number of tables. Given n checkpoints $c_1 < c_2 < \cdots < c_n$, the expected number of f invocations that can be removed through checkpoints is*

$$-\frac{mt^2l}{N}\sum_{j=1}^{n}\{\int_{c_j/t}^{c_{j+1}/t}(1-x)e^{-\frac{mtlx}{N}}(\frac{mt}{2N}\cdot x-1)\,dx\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}}{2^{j-u}}\}$$
$$+\frac{m^2t^2l}{4N^2}\cdot\sum_{j=1}^{n}\{\int_{c_j/t}^{c_{j+1}/t}(1-x)e^{-\frac{mtlx}{N}}\,dx\cdot\sum_{u=0}^{j-1}\frac{c_{u+1}^2}{2^{j-u}}\}.$$

B Maple Code for Optimal Checkpoints in Perfect Rainbow Tradeoff

```
> with(Optimization);
> N := 2^24;
> m := 65536;
> t := 300;
> l := 3;
> n := 4;
> c := array(1 .. n);
> S := {}:
> for i from 1 to n-1 do
> S := S union {c[i] <= c[i+1]}
> end do:
> S := S union {c[n] <= t}:
> eval(
> -m^2*l*t^3/N/N/2*sum(int((1-x)*x*exp(-m*t*l/N*x),
> x=c[j]/t..c[j+1]/t)*sum(c[u+1]*2^(u-j),u=0..j-1),
> j=1..n-1)+m*l*t^2/N*sum(int((1-x)*exp(-m*t*l/N*x),
> x=c[j]/t..c[j+1]/t)*sum(c[u+1]*2^(u-j),u=0..j-1),
> j=1..n-1)+m^2*t^2*l/4/N/N*sum(int((1-x)*exp(-m*t*l/N*x),
> x=c[j]/t..c[j+1]/t)*sum((c[u+1])^2*2^(u-j),u=0..j-1),
> j=1..n-1)):
> Maximize(%,S, assume=nonnegative);
```