Improvement and Analysis of VDP Method in Time/Memory Tradeoff Applications

Wenhao Wang[1,2], Dongdai Lin[1], Zhenqi Li[1,2], and Tianze Wang[1,2]

[1] SKLOIS, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] Graduate University of Chinese Academy of Sciences, Beijing, China
{wangwenhao,ddlin,lizhenqi,wangtz83}@is.iscas.ac.cn

Abstract. In many cases, cryptanalysis of a cryptographic system can be interpreted as the process of inverting a one-way function. TMTO is designed to be a generic approach that can be used on any one-way function independent of the structure of the specific target system. It was first introduced to attack block ciphers by Hellman in 1980. The distinguished point (DP) method is a technique that reduces the number of table look-ups performed by Hellman's algorithm. A variant of the DP (VDP) method is introduced to reduce the amount of memory required to store the pre-computed tables while maintaining the same success rate and online time. Both the DP method and VDP method can be applied to Hellman tradeoff or rainbow tradeoff.

We carefully examine the technical details of the VDP method and find that it is possible to construct functions for which the original method fails. Based on the analysis, we propose a modification of the VDP method. Furthermore, we present an accurate version of the tradeoff curve that does not ignore the effect of false alarms and takes storage reduction techniques into consideration. We find optimal parameter sets of this new method by minimizing the tradeoff coefficient. A more exact and fair comparison between tradeoff algorithms is also given, which shows that our method applied to the Hellman tradeoff performs best among them.

1 Introduction

The objective of a time memory tradeoff (TMTO) algorithm is to do some one-time work so that each time the algorithm is executed it is more efficient. TMTO method can be applied to numerous cryptosystems. In 1980, Hellman described a chosen plaintext attack against the Data Encryption Standard, which is the first time memory tradeoff attack for block ciphers [3]. Let $E_k()$ be the encryption function of the block cipher. A fixed message msg is chosen and a one-way function f is defined from keys to ciphertexts by $f(k) = E_k(msg)$. The task of the cryptanalyst is to obtain k given msg and the corresponding ciphertext. This can be seen as inverting a one-way function f and later work has viewed TMTO as a general one-way function inverter.

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

© Springer-Verlag Berlin Heidelberg 2011

plaintext. In an off-line phase, a set of tables are constructed. The tables store keys and an encryption of msg under an unknown key. In the online phase, the goal is to find the unknown key by making use of these pre-computed tables. The main idea of Hellman tradeoff is to store only part of the tables. This incurs a cost in the online phase and leads to a tradeoff between the memory and online time requirements.

After the inspiring work of Hellman, several articles have dealt with time memory tradeoff. In 1982, Rivest suggested an optimization based on distinguished points (DP) which greatly reduces the amount of look-up operations needed to detect a matching end point. Simply put, a DP is a point in the one-way chain that satisfies a preset condition. Chains are not generated with a given length but they stop at the first occurrence of a distinguished point. During the online phase, a look-up is carried out only when a distinguished point appears. Because merging chains significantly degrade the efficiency of the tradeoff, Borst, Preneel, and Vandewalle suggested in 1998 to clean the tables by discarding the merging and cycling chains [1]. This new kind of tables, called perfect table, substantially decreases the required memory. Later, Standarert dealt with a more realistic analysis of distinguished points in [9].

In 2003, Oechslin introduced the tradeoff based on rainbow tables, which reduces the TMTO attack complexity by a factor of 2 [8]. A rainbow table uses a different reduction function for each column of the table. Thus two different chains can merge only if they have the same key at the same position of the chain. This makes it possible to generate much larger tables. Up until now, tradeoff techniques on rainbow tables are the most efficient ones. In 2008, J. Hong proposed a variant of the DP technique, named variable DP (VDP) [4]. It completely removes the need to store the start points, as the chain end contains information about the chain beginning.

We carefully examine the details of J. Hong's method and find that it is possible to construct functions for which the original method fails. In fact, Fiat and Naor showed that there exists functions which are polynomial time indistinguishable from a random function and for which Hellman's attack fails [2]. We show that a Fiat-Naor type counterexample works for the VDP method. This leads us to the question of figuring out a method to obtain uniformly distributed start points. We propose a modification of the VDP method by suggesting the use of a generic diffusion function for generating start points. Furthermore, we present an accurate version of the tradeoff curve that does not ignore the effect of false alarms and takes storage reduction techniques into consideration. We find optimal parameter sets of this new method by minimizing the tradeoff coefficient. A more exact and fair comparison between tradeoff algorithms is also given, which shows our method applied to the Hellman tradeoff performs best among them.

The rest of the paper is organized as follows. We start by briefly reviewing some of the previous TMTO works. We provide in Section 3 our improvement over VDP method. In Section 4 we apply this new VDP method to both Hellman tradeoff and rainbow tradeoff. A brief comparison of previous TMTO algorithms is given in Section 5. We end the paper with a short conclusion in Section 6.

2 Previous Works

Let $f : \mathcal{N} \rightarrow \mathcal{N}$ be the one-way function to be inverted and we always suppose that $\mathcal{N} = \{0, 1\}^n$ and $N = 2^n$. As a rule, the time memory tradeoff methodology tries to invert an arbitrary one-way function and we will follow this approach.

All tradeoff algorithms consist of a pre-computation phase and an online phase. In the off-line phase, the attacker constructs tables that contain possible keys; this process is approximately equivalent to the exhaustive search however only a part of the tables are stored. If the attack is only to be executed once then an exhaustive key search would be more efficient. But if the attack is to be conducted many times, the pre-computation work can be amortized. During the online phase, the attacker expects to recover the key from the pre-computed tables. This leads to a tradeoff between the memory required in the off-line stage and time required in the online phase.

2.1 Hellman Tradeoff

During the pre-compute phase we build m Hellman chains with the form($1 \leq j \leq m$): $SP_j = X_{j,1} \xrightarrow{f} Y_{j,1} \xrightarrow{r} X_{j,2} \xrightarrow{f} Y_{j,2} \xrightarrow{r} \cdots \xrightarrow{f} Y_{j,t} \xrightarrow{r} EP_j$. Then we discard all intermediate points of the Hellman chains and just pairs of start points and ending points $\{(SP_j, EP_j)\}_{j=1}^{m}$ are stored in one table. The stored pairs are sorted with respect to the ending points. In practice we suppose l tables are built. A different reduction function r_i is used in each table, and we denote $r_i(f(x))$ by $f_i(x)$.

In the online phase, given y_0, it is required to find x_0 satisfying $y_0 = f(x_0)$. We demonstrate the search for x_0 in the i-th table as follows.

Recursively we compute $Y_0 = r_i(f(x_0)) = f_i(x_0)$ and $Y_k = f_i(Y_{k-1})$, where $k = 1, 2, \cdots, t-1$. After each Y_k is obtained, check if it appears as an ending point in the table. Whenever a match $Y_k = EP_j$ is found, we compute $x = X_{j,t-k-1} = f_i^{t-k-1}(SP_j)$. There is a large chance that $f_i(x) = Y_0$, which is equivalent to $f(x) = f(x_0)$, due to r_i being injective. In such a case, the algorithm returns the correct x. But, as f_i^k is not injective, there could be a merge between the Hellman chains and online chains, and it is possible to have $f(x) \neq f(x_0)$. This is referred to as a false alarm.

2.2 Rainbow Tradeoff

Rainbow tradeoff is proposed by Oechslin to reduce the online time cost and the probability that a merge appears in a table. Instead of using the same reduction function f_i for each table, t different reduction functions are used to generate different columns in a single table.

A rainbow chain is build with the form: $SP_j = X_{j,1} \xrightarrow{f} Y_{j,1} \xrightarrow{r_1} X_{j,2} \xrightarrow{f} Y_{j,2} \xrightarrow{r_2} \cdots \xrightarrow{f} Y_{j,t} \xrightarrow{r_t} EP_j$. In the online phase, we apply r_t to the ciphertext and look up the result in the endpoints of a table. If a match is found, we can rebuild the chain from the corresponding start point. Otherwise, we continue to apply r_{t-1}, f to the ciphertext. The total calculations of the rainbow method is half that of the Hellman method.

2.3 Distinguished Points Method

Distinguished points are keys that satisfy a given criterion, e.g., the last k bits are all zero. The crucial insight is that we don't use fixed-length chains, and instead, we simply construct a chain until some easily distinguished point is found. In practice, we would want to set a limit on the maximum length of a chain as $\hat{t}$ and reject any chain that exceeds the limit. Instead of looking up in the table each time a key is generated, a look-up is carried out in the table only when the online chain reaches a distinguished point.

2.4 Variable Distinguished Points Method

The main objective of the VDP method is to eliminate the need to store the start points. Just as the original DP method, generation of a Hellman chain is continued until a point satisfying a certain condition is found. We allow this condition to depend on the start point of the chain and the ending point contains information about the corresponding start point. We are able to recover the start point from the ending point.

First, we restrict to the typical parameters $m = t = N^{\frac{1}{3}}$ and set $d = \frac{1}{3}log_2 N$. When generating the j-th chain of the i-th table. Set the start point $SP_j^i = (0 \parallel i \parallel j)$, where each of the three concatenated components are of d bits. Iteratively we generate a Hellman chain by $X_{j,k} = f_i(X_{j,k-1})$. It is terminated only when the most significant d bits of some $X_{j,k}$ is found to be j. Chains longer than a preset $\hat{t}$ are discarded. Just the last $2d$ bits of the ending point in the j-th chain is stored in the j-th position of the i-th table and no table sorting is involved. Storing chain length information would reduce online time spent dealing with false alarms, but this is not mandatory. The online phase of the VDP method is same as the DP method. When a match is found, we can recover the corresponding start point using the table number and the position index.

3 Improvement of VDP Method

In this section, we first present a counter-example for which the VDP method fails. This is similar to the example given by Fiat and Naor for the Hellman tradeoff. Then we give the description of our improvement over VDP method.

3.1 Construction of a Counter-Example to VDP Method

For both Hellman and rainbow method to work, the functions f_i's need to be pairwise unrelated. This requirement for the Hellman method was carefully examined by Fiat and Naor [2]. Sourav Mukhopadhyay examined the requirements for rainbow method and showed that the counter based method to generate the iterate functions makes it possible to construct one-way functions for which both the Hellman tradeoff and rainbow tradeoff fail [7].

Start points of the i-th table in the original VDP method are chosen with the form $(0 \parallel i \parallel j)$. They are not uniformly distributed and the original VDP method suffers from a problem similar to the one described by Fiat and Naor for the Hellman method. Denote $\frac{n}{3}$ by d for convenience. Now we can construct a function f as follows: $f : \{0,1\}^n \to \{0,1\}^n$ with the property that for any $x = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$, if $x_1 = x_2 = \cdots x_d = 0$ and $x_{2d+d_0} = x_{2d+d_0+1} = \cdots x_{3d} = 0$, then $f(x) = 0$. Map of the rest x's induces a permutation. We may construct a cryptographic scheme considering such a function f by choosing d_0 properly to make f perform like a random function. For a VDP table, huge number of start points map into zeroes, leaving a large number of merges inside a table. This will lead to the failure of the method.

3.2 Main Idea of Our Improvemet

Based on the analysis, we present a modification of the VDP method and try to invert any one-way function. We would prefer to treat the start points as a univariant function of j over $\mathcal{N}$, denoted by $sp(j)$, where j is the chain index. Before the description of our improvement, we first provide the requirements of the function sp.

1. Given the chain index j, we obtain the corresponding start point by an invocation of the function sp. The invocation is called each time we intend to generate a new chain during the pre-computation phase or we find a match of the ending point during the online phase. It is strongly recommended that the function can be efficiently computed to reduce both the pre-computation time and the online time.
2. As mentioned above, output of sp needs to distribute uniformly over $\mathcal{N}$. This ensures that it is not possible to construct a Fiat-Naor type example prior for the new VDP method.
3. Obviously, start points should be chosen pairwise not equal. It precludes the possibility that chains merge from the very beginning. This can be easily ensured by choosing injective functions as sp.
4. Referring to flexibility, the discussion is focused on the generation of a series of function sp's. Parameters of the new VDP method are being thoroughly analysed in the following section. Selection of sp will put restriction on the choice of parameters in our method. Enough flexibility leaves us room to optimize the new VDP method.

5. As we need to generate nearly up to $2^{\frac{n}{3}}$ chains in each table, that means we need at least $2^{\frac{n}{3}}$ different start points. In terms of the cases that different start points map into same images, image set of sp is required to cover most of the range.

A univariant function filling the above requirements will hereinafter to be referred as a generic diffusion function. As the computation of power of an integer can be time-consuming, we suggest using linear functions with the form $sp(j) = x_i + k \cdot j$, $1 \le i \le r$ and $0 \le j \le s-1$ as start point while generating the j-th chain in a VDP table. When we create a new table in the off-line phase, we fix r and s and choose r different x_i's. Each x_i can generate up to s different start points. We will show that linear functions is sufficient. Uniform distribution and good coverage are easy to reach. The flexibility lies in the fact that we choose x_i and j freely only if the following equations hold (injection): $\gcd(k, 2^n) = 1$, $x_{i_1} + k \cdot j_1 \ne x_{i_2} + k \cdot j_2 \pmod{2^n}$ for different i_1, i_2 and j_1, j_2.

Let k be an integer not less than r and k has no common factor with N. x_i's belong to different congruence classes modulo k. We obtain an improvement over VDP method. If we choose 1 as k, the table index multiplies $2^{\frac{n}{3}}$ as x_1 and $2^{\frac{n}{3}}$ as s, we obtain the original VDP method. If we choose $2^n - 1$ as k, $2^{\frac{n}{3}}$ as r and 1 as s, we obtain the original DP method. We emphasize again that this flexility makes it possible to optimize the new method and seek better performance.

4 Applying the New Method to Hellman Tradeoff and Rainbow Tradeoff

4.1 Applying the New VDP Method to Hellman Tradeoff

When creating a new table, we choose r different x_i's belonging to different congruence classes modulo k. For each x_i, we generate s start points with the form

$$sp(i,j) = x_i + k \cdot j,\ 0 \le j \le s-1.$$

The new VDP chains applied to Hellman tradeoff are of the form:

$$\begin{aligned}
&sp(1,0) = X_{1,1} \xrightarrow{f} Y_{1,1} \xrightarrow{r} X_{1,2} \xrightarrow{f} Y_{1,2} \xrightarrow{r} \cdots \xrightarrow{f} Y_{1,t} \xrightarrow{r} EP_1 \\
&\quad\vdots \\
&sp(1,s-1) = X_{s,1} \xrightarrow{f} Y_{s,1} \xrightarrow{r} X_{s,2} \xrightarrow{f} Y_{s,2} \xrightarrow{r} \cdots \xrightarrow{f} Y_{s,t} \xrightarrow{r} EP_s \\
&sp(2,0) = X_{s+1,1} \xrightarrow{f} Y_{s+1,1} \xrightarrow{r} X_{s+1,2} \xrightarrow{f} Y_{s+1,2} \xrightarrow{r} \cdots \xrightarrow{f} Y_{s+1,t} \xrightarrow{r} EP_{s+1} \\
&\quad\vdots \\
&sp(2,s-1) = X_{2s,1} \xrightarrow{f} Y_{2s,1} \xrightarrow{r} X_{2s,2} \xrightarrow{f} Y_{2s,2} \xrightarrow{r} \cdots \xrightarrow{f} Y_{2s,t} \xrightarrow{r} EP_{2s} \\
&\quad\vdots
\end{aligned}$$

The $((i-1)s + j + 1)$-th chain ends with EP only when the most significant bits of EP is found to be $(i \parallel j)$. The ending points we have reached are stored sequentially. We have to store x_i's but neither the start points nor the first $\lceil log_2(s)\rceil + \lceil log_2(r)\rceil$ bits of the ending points need to be stored. In the online phase when we want to check whether a point from the online chain is an ending point, we determine i and j from the first $\lceil log_2(s)\rceil + \lceil log_2(r)\rceil$ bits of the point and compute the corresponding index of the off-line table by $index = j+(i-1)\cdot s$. No searching is involved. If a match is found, the corresponding start point can be recovered by $sp(i,j) = x_i + k\cdot j$.

There remains a question of choosing proper r and s to optimize the new VDP method. We first present the result of optimal parameters in DP tradeoff. We do not store chain length information. As is expected, it increases the effort in resolving false alarms. But it turns out that the increase in total computation is minimal in comparison to the reduction in storage. We denote t, m, $\hat{t}$ and l the parameters of the tradeoff, which are respectively the average length of the chains constituting the tables, the number of chains per table, the maximum chain length in the table and the number of tables.

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

$$mt^2 = 1.47N,\ \ l = 1.25ln(\frac{1}{1-D_{ps}})t,\ \ and\ \hat{t} = 1.97t$$

attains the given value D_{ps} as its probability of success, and exhibits tradeoff performance corresponding to

$$D_{tc} = 6.5452D_{ps}\{ln(1-D_{ps})\}^2$$

as the four parameters are varied. The time memory tradeoff curve for the DP tradeoff is $TM^2 = D_{tc}N^2$.

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

$$D_{pc} = 1.8349ln(\frac{1}{1-D_{ps}})$$

in multiples of N.

The three relations restricting the parameter choices give optimal parameters in the sense that no choice of m, t, l, and $\hat{t}$ can lead to a tradeoff coefficient smaller than the above while achieving D_{ps} as its probability of success. The storage size M appearing in the above tradeoff curve is the total number of start point and ending point pairs that need to be stored in the tradeoff tables. Proof of Proposition 1 is similar to that of Proposition 35 in [5]. We exhibit it in Appendix A.

Noting that performance of the new VDP method and DP method differs only in storage if we disregard the time spent on table look-ups, we find optimal parameters of our method by applying Proposition 1.

Proposition 2. *Given N and D_{ps}, the optimal parameters of the new VDP tradeoff are subjected to the following relations*

$$rs = 1.20N^{\frac{1}{3}},\ l = \ 1.49ln(\frac{1}{1-D_{pc}}) \cdot N^{\frac{1}{3}},\ \hat{t} = 2.35N^{\frac{1}{3}},\ \textit{and}\ k = N^{\frac{1}{3}} + 1.$$

The tradeoff coefficient for the new VDP method under the above optimal parameters is

$$D_{tc} = 0.7384D_{ps}\{ln(1-D_{ps})\}^2.$$

The number of one-way function invocations required for the pre-computation phase is

$$D_{pc} = 1.8349ln(\frac{1}{1-D_{ps}})$$

in multiples of N.

Proof. To apply Proposition 1 to our new VDP method, we first update the tradeoff parameters. If a chain is generated with a random function, with the chain length bound set to $\hat{t}$, the probability of not obtaining a DP chain will be $(1-\frac{1}{t})^{\hat{t}} \approx e^{-\frac{\hat{t}}{t}}$. During the pre-computation phase we generate rs distinct start points, so after discarding chains not reaching a distinguished point, the expected number of chains is $rs(1-e^{-\frac{\hat{t}}{t}})$. Chains end only when some point with its most significant bits being $(i \parallel j)$ is found. Thus the average length of the chains is $2^{\lceil log_2(s)\rceil + \lceil log_2(r)\rceil}$. Approximately we replace it by rs. So we get $t = rs$.

Now referring to Proposition 1, the relations may equivalently be stated as

$$rs(1-e^{-\frac{\hat{t}}{t}}) \cdot (rs)^2 = 1.47N,\ l = 1.25ln(\frac{1}{1-D_{ps}})rs,\ \text{and}\ \hat{t} = 1.97t.$$

Then

$$rs = 1.20N^{\frac{1}{3}},\ l = \ 1.49ln(\frac{1}{1-D_{pc}}) \cdot N^{\frac{1}{3}},\ \hat{t} = 2.35N^{\frac{1}{3}}.$$

As only the last $\frac{2}{3}n$ bits of the ending points are stored, the new VDP method saves the memory cost by a factor of $\frac{1}{3}$. We replace M in the tradeoff curve of Proposition 1 by $\frac{M}{3}$ and obtain tradeoff curve of the new VDP method. The time-memory tradeoff curve of the new VDP method is $TM^2 = D_{tc}N^2$, where

$$D_{tc} = 0.7384D_{ps}\{ln(1-D_{ps})\}^2.$$

Recall that k is greater than r and coprime with N. We simply assign k a value of $N^{\frac{1}{3}} + 1$. □

Before continuing, note that we always suppose $m = t$ in the VDP method. In such a case we reach the best tradeoff performance in theory. When we want parameters m and t to differ by a large factor or when m and t are not powers of 2, we may need to convert Proposition 2 to other forms through applying Proposition 1.

4.2 Extending the New VDP Method to Rainbow Tradeoff

As mentioned above, the rainbow method applies a different f_i to every column. During the online phase, given a target $f(x_0)$, it is necessary to know the chain length to decide which f_i to use. Results for DP method applied to rainbow tradeoff are listed below, and they are easy to be extended to the new VDP method. In this section, we denote by t the inverse of the probability to find a distinguished point and set $\hat{t} = ct$.

During the pre-compution phase, we first choose $\widetilde{m}_0$ start points and use different iteration f_i's for different columns. We denote the number of different keys in column i by $\widetilde{m}_i$. When $0 \le i < \hat{t}$, as about $\widetilde{m}_i \cdot \frac{1}{t}$ points reach distinguished points, we can easily find out a recurrence relation on $\widetilde{m}_i$: $\widetilde{m}_{i+1} = N(1 - (1 - \frac{1}{N})^{\widetilde{m}_i(1-\frac{1}{t})})$, which can be approximated by $\widetilde{m}_{i+1} = \widetilde{m}_i(1 - \frac{1}{t})$. So we get $\widetilde{m}_i = \widetilde{m}_0(1 - \frac{1}{t})^i$.

Then we discard all chains not reaching a DP in length $\hat{t}$. The number of different keys in column i after this process is denoted by m_i. Thus

$$m_i = \widetilde{m}_i - \widetilde{m}_{\hat{t}}(1 - \frac{1}{t}) = \widetilde{m}_0(1 - \frac{1}{t})^i - \widetilde{m}_0(1 - \frac{1}{t})^{\hat{t}+1}. \tag{1}$$

Following this, the rainbow table is sorted first with respect to chain lengths and then with respect to the ending points within those chains of same length. The start point, the ending point and the chain length are stored. The total number of chains that need to be stored is $m_0 = \widetilde{m}_0(1 - e^{-c})$.

During the online phase, we assume that the l tables are processed with the simultaneous approach, which means the chains of same length in different tables are searched before moving to the next column.

Probability of success. The number of distinct nodes expected in each column is given by the stated m_i. So the probability for the first k iterations of the online phase to fail, denoted by P_k, is

$$\prod_{i=1}^{k-1}(1 - \frac{m_{\hat{t}-i}}{N})^l.$$

We have ignored the interdependence between columns and approximate it by

$$\begin{aligned} P_k &= \textstyle\prod_{i=1}^{k-1}(1 - \frac{m_{\hat{t}-i}}{N})^l \approx \prod_{i=1}^{k-1} e^{-\frac{m_{\hat{t}-i}l}{N}} &&= e^{-\frac{l\sum_{i=1}^{k-1} m_{\hat{t}-i}}{N}} \\ &= e^{-\frac{l}{N}\sum_{i=1}^{k-1}[(\widetilde{m}_0(1-\frac{1}{t})^{\hat{t}-i} - \widetilde{m}_0(1-\frac{1}{t})^{\hat{t}})]} &&= e^{-\frac{l\widetilde{m}_0 t}{N}(e^{\frac{k-1}{t}-c} - e^{-c} - \frac{k}{t}\cdot e^{-c})}. \end{aligned}$$

We already know that the pre-computation requirement of DP method is m_0tl regardless of the chain length bound and we set $RN = m_0tl$, where R means the pre-computation coefficient. We finally yield

$$P_k = e^{-\frac{R}{e^c-1}(e^{\frac{k-1}{t}} - 1 - \frac{k}{t})}. \tag{2}$$

So, the success rate of DP method applied to rainbow tradeoff is

$$P = 1 - P_{\hat{t}} = 1 - e^{-\frac{R}{e^c-1}(e^c - 1 - c)}. \tag{3}$$

Tradeoff curve. We first provide the result of expected numbers of false alarms associated with a single rainbow matrix at the i-th iteration, denoted by $E_{fa}(i)$. A collision happens with probability of $\frac{1}{N}$ under an invocation of f. During the i-th iteration, for a chain of length $\dot{t}$, f is computed $\dot{t}-(\hat{t}-i)$ times. The probability that a false alarm occurs is $1-(1-\frac{1}{N})^{\dot{t}-(\hat{t}-i)}$, which can be approximated by $\frac{\dot{t}-(\hat{t}-i)}{N}$. Thus, taking false alarms associated with all rows into consideration,

$$\begin{aligned} E_{fa}(i) &= \textstyle\sum_{j=\hat{t}-i+1}^{\hat{t}}(m_j - m_{j+1}) \cdot \frac{j-(\hat{t}-i)}{N} = \frac{1}{N}\sum_{j=\hat{t}-i+1}^{\hat{t}} \widetilde{m}_0(1-\frac{1}{t})^j \cdot \frac{1}{t} \cdot (j-\hat{t}+i) \\ &= \textstyle\frac{\widetilde{m}_0 t}{N}[(1-\frac{1}{t})^{\hat{t}-i+2} \cdot (1-(1-\frac{1}{t})^{i-1}) - \frac{i}{t}(1-\frac{1}{t})^{\hat{t}+1})]. \end{aligned}$$

Hence, the expected total running time can be written as

$$\begin{aligned} T &= \textstyle\sum_{i=1}^{\hat{t}} l[(i-1)+(\hat{t}-i+1)\cdot E_{fa}(i)] \cdot P_i \\ &= \textstyle t^2 l \sum_{i=1}^{\hat{t}}[\frac{i-1}{t} + \frac{\hat{t}-i+1}{t} \cdot E_{fa}(i)]e^{-\frac{R}{e^c-1}(e^{\frac{i-1}{t}}-1-\frac{i}{t})} \cdot \frac{1}{t}. \end{aligned}$$

This may be approximated by the definite integral

$$T = t^2 \int_0^c [lu + (c-u)\cdot \frac{R}{e^c-1}(e^u - 1 - u)]e^{-\frac{R}{e^c-1}(e^u-1-u)}du. \tag{4}$$

It now suffices to combine this with the storage size $M = m_0 l$ and obtain the tradeoff curve $TM^2 = DN^2$, where the tradeoff coefficient is

$$D = R^2 \textstyle\int_0^c [lu + (c-u)\cdot \frac{R}{e^c-1}(e^u - 1 - u)]e^{-\frac{R}{e^c-1}(e^u-1-u)}du. \tag{5}$$

Optimal parameters. Recalling that a smaller tradeoff coefficient implies better tradeoff performance, we show next how to choose optimal parameters l, P and c to minimize the tradeoff coefficient. As a much smaller parameter l is used in rainbow tradeoff, we treat it as discrete numbers $1, 2, 3, 4$ and so on. For a fixed l and R, we plot the tradeoff coefficient D to display the relationship between D and P. Results are shown in Figure 1.

Using numerical method, we find out that the tradeoff coefficient increases strictly with the increasing of l, c or P. It is obvious to choose 1 as l and a smallest c to obtain the best tradeoff performance. However, the tradeoff curve does not take pre-computation time into account. As a matter of fact the above optimal tradeoff is achieved at the cost of high pre-computation time. According to Equation 3,

$$R = -\frac{ln(1-P)\cdot(e^c-1)}{e^c-1-c}. \tag{6}$$

In practice, the pre-computation coefficient R is not too large. We numerically solve Equation 5 and results are shown in Table 1. We explain the content of the table with examples. Suppose one aims to achieve the success probability of 0.85, and the pre-computation time is upper bounded by $3N$. It is optimal to build one DP-Rainbow table and set $\hat{t}$ equals $1.75t$. The tradeoff coefficient is expected to be around 8.7968.

As chain length information is stored in our new VDP method when applied to rainbow tradeoff, it saves the memory cost by a factor of $\frac{3}{4}$. It follows that:

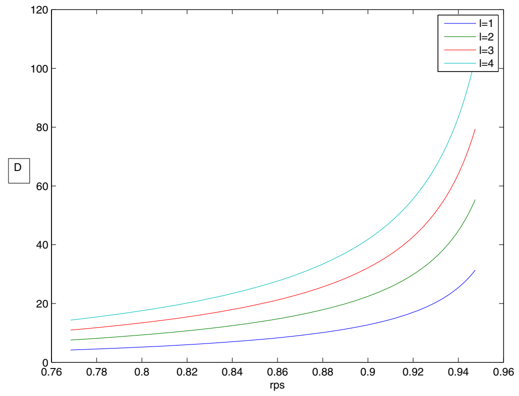

Fig. 1. Tradeoff coefficient D with different P and l, R = 3.5

Proposition 3. *Let $0 < P < 1$ be any given fixed value. The pre-computation time is upper bounded by RN. Locate c in the above table. Then the new VDP method applied to rainbow tradeoff with optimal parameters satisfying*

$$l = 1,\ rs = \sqrt{\frac{RN}{1-e^{-c}}},\ \hat{t} = c\sqrt{\frac{RN}{1-e^{-c}}},\ k = \sqrt{N}+1$$

attains the given value P as its success probability. The tradeoff coefficient is

$$D = \frac{9R^2}{16}\int_0^c [lu + (c-u)\cdot\frac{R}{e^c-1}(e^u-1-u)]e^{-\frac{R}{e^c-1}(e^u-1-u)}du.$$

5 Comparison

In this section, we compare our new VDP method with previous tradeoff algorithms using tradeoff curves. It should first be noticed that a better tradeoff coefficient should always be achievable, if one decides to sacrifice the success probability for finding the correct password. A fair comparison of the algorithms should compare them at the same success rate. We summarize the relevant facts in Table 2.

The presented time complexity T of the first half of the table disregards false alarms and uses general parameters satisfying matrix stopping rule. The second half of the table deals with real world cracking, which takes false alarms into consideration. The optimal parameters are obtained at the given success rate. From the table, we can clearly figure out that our new method applied to Hellman table achieves a better performance. As to the method applied to rainbow tradeoff, chains are sorted with respect to the chain lengths and we have to store chain length information. It doesn't bring us a significantly better result.

Table 1. Optimal parameters and corresponding tradeoff coefficient for a given P

l	P	R	c	tradeoff coefficient	l	P	R	c	tradeoff coefficient
1	0.7	2.8802	1.00	2.9915	1	0.75	3.3163	1.00	3.8091
1	0.8	3.8501	1.00	4.8997	1	0.85	4.5383	1.00	6.4366
1	0.9	5.5083	1.00	8.8241	1	0.95	7.1664	1.00	13.4272
1	0.7	2.5	1.2	3.2549	1	0.7	3	1.0	2.9915
1	0.75	2.5	1.45	4.5906	1	0.75	3	1.13	3.9939
1	0.8	2.5	1.8	6.8245	1	0.8	3	1.38	5.7169
1	0.85	2.5	2.39	11.6141	1	0.85	3	1.75	8.7968
1	0.9	2.5	3.92	31.0272	1	0.9	3	2.44	16.3704
1	0.95	2.5	-	-	1	0.95	3	8.49	180.7163
1	0.7	3.5	1.0	2.9915	1	0.7	4	1.0	2.9915
1	0.75	3.5	1.0	3.8091	1	0.75	4	1.0	3.8091
1	0.8	3.5	1.13	5.1822	1	0.8	4	1.0	4.8997
1	0.85	3.5	1.4	7.5863	1	0.85	4	1.17	6.8800
1	0.9	3.5	1.86	12.6852	1	0.9	4	1.52	10.9405
1	0.95	3.5	3.12	34.1475	1	0.95	4	2.33	24.0022

Table 2. Comparison of tradeoff algorithms, denote success rate by P

	parameter selection	Pre-computation time:$\times$ N	tradeoff curve
Hellman	$mt^2 = N$	1	$TM^2 = N^2$
Rainbow	$mt^2 = N$	1	$TM^2 = \frac{1}{2}N^2$
Hellman+DP	$mt^2 = N$	-	$TM^2 = N^2$
Hellman+VDP	$mt^2 = N$, $\hat{t} = ct$	-	$TM^2 = cN^2$
Rainbow+DP	$mt^2 = N$, $\hat{t} = ct$	-	$TM^2 = \frac{c^2}{2}N^2$
Rainbow+VDP	$mt^2 = N$, $\hat{t} = ct$	-	$TM^2 = \frac{c^2}{2}N^2$
Hellman	Optimal, $P = 80\%$	2.1733	$TM^2 = 3.11N^2$
Rainbow	Optimal, $P = 80\%$	1.9814	$TM^2 = 2.20N^2$
Hellman+DP	Optimal, $P = 80\%$	2.9532	$TM^2 = 13.77N^2$
Rainbow+DP	Optimal, $P = 80\%$	3	$TM^2 = 5.72N^2$
Hellman+new VDP	Optimal, $P = 80\%$	2.9532	$TM^2 = 1.53N^2$
Rainbow+new VDP	Optimal, $P = 80\%$	3	$TM^2 = 3.22N^2$
Hellman	Optimal, $P = 90\%$	3.1093	$TM^2 = 7.17N^2$
Rainbow	Optimal, $P = 90\%$	2.8068	$TM^2 = 4.68N^2$
Hellman+DP	Optimal, $P = 90\%$	4.2250	$TM^2 = 31.68N^2$
Rainbow+DP	Optimal, $P = 90\%$	3.5	$TM^2 = 12.69N^2$
Hellman+new VDP	Optimal, $P = 90\%$	4.2250	$TM^2 = 3.52N^2$
Rainbow+new VDP	Optimal, $P = 90\%$	3.5	$TM^2 = 7.14N^2$

6 Conclusion

In this paper, we present a rigorous analysis of the VDP method and suggest using a generic diffusion function to generate start points. Optimal parameters of the new VDP method are also given. Furthermore, we give a careful analysis on how to apply this method to rainbow tradeoff. We present an accurate version of the tradeoff curve that does not ignore the effect of false alarms and takes storage reduction techniques into consideration. A fair comparison of all previous tradeoff algorithms are presented, which shows that our method applied to Hellman tradeoff performs best among them.

The optimal parameters are determined by just minimizing the tradeoff coefficient. However, parameters achieving better tradeoff performance may require more pre-computation, and with large scale implementations of the tradeoff technique, lowering the pre-computation cost may be significantly more valuable than achieving better tradeoff performance. Our analysis may seem interesting in view of optimal usage of tradeoff algorithms, but can be of limited value in practice. One should consider our work as a guide and use it to arrive at their final judgements.

References

1. 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)
2. Fiat, A., Naor, M.: Rigorous time/space tradeoffs for inverting functions. In: Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing, pp. 534–541. ACM (1991)
3. Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), 401–406 (1980)
4. Hong, J., Jeong, K.C., Kwon, E.Y., Lee, I.-S., Ma, D.: Variants of the Distinguished Point Method for Cryptanalytic Time Memory Trade-offs. In: Chen, L., Mu, Y., Susilo, W. (eds.) ISPEC 2008. LNCS, vol. 4991, pp. 131–145. Springer, Heidelberg (2008)
5. Hong, J., Moon, S.: A comparison of cryptanalytic tradeoff algorithms. Technical report, Cryptology ePrint Archive, Report 2010/176 (2010)
6. Kim, I.J., Matsumoto, T.: Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 82(1), 123–129 (1999)
7. Mukhopadhyay, S., Sarkar, P.: Application of lFSRs in Time/Memory Trade-off Cryptanalysis. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 25–37. Springer, Heidelberg (2006)
8. 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)
9. Standaert, F.X., Rouvroy, G., Quisquater, J.J., Legat, J.D.: A Time-memory Tradeoff Using Distinguished Points: New Analysis & FPGA Results. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–30. Springer, Heidelberg (2003)
10. Thing, V.L.L., Ying, H.M.: A novel time-memory trade-off method for password recovery. Digital Investigation 6, S114–S120 (2009)

A Proof of Proposition 1

We first fix notation that is used in our analysis of DP tradeoff.

m**:** number of chains in a single DP table
t**:** the DP property is satisfied with a probability of $\frac{1}{t}$
D_{msc}**:** the matrix stopping constant, $mt^2 = D_{msc}N$, $D_{msc} = \Theta(1)$
l**:** number of DP tables, $l = \Theta(t)$
$\hat{t}$**:** chain length bound, $\hat{t} = ct$, $c = \Theta(1)$

Proposition 4. *[5] The pre-computation phase of the DP tradeoff is expected to require mtl one-way function invocations. Denote the pre-computation coefficient by $D_{pc} = \frac{mtl}{N}$. The success probability of the DP tradeoff is*

$$D_{ps} = 1 - \big(1 - \frac{D_{cr}mt}{N}\big)^l \approx 1 - exp\big(- D_{cr}\frac{mtl}{N}\big) = 1 - e^{-D_{cr}D_{pc}}.$$

D_{cr} is the coverage rate of a DP table, and

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

where $\Xi = \sqrt{1 + \frac{2D_{msc}}{1-e^{-\hat{t}/t}}}$.

Proposition 5. *[5] Fix a random function $f : N \to N$ and suppose that we are given a pre-computed DP chain of length $j < \hat{t}$, generated with f from a random non-DP start point. If a second chain is generated with f from a random start point, the probability for it to become a DP chain of length i and merge with the given pre-computed chain is*

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

Lemma 1. *The number of extra one-way function invocations induced by alarms is expected to be*

$$t\frac{D_{msc}}{1 - e^{-c}}(-2 + c + 2ce^{-c} + e^{-c} + e^{-2c} - \frac{c^2}{2}e^{-c})$$

for each DP table.

Proof. Proof is same as that in [5] except that we do not store the chain length information, which causes extra effort in dealing with false alarms.

Recall the probability for a random chain to become a DP chain within the chain length bound $\hat{t}$ is $1 - e^{-\hat{t}/t}$. Rather than requiring each table to contain exactly m entries, we assume that each pre-computation DP table is always generated from $m_0 = \frac{m}{1-e^{-\hat{t}/t}}$ distinct start points. Then we can expect to collect approximately m chains that terminate at DPs.

When the chains are generated from m_0 non-DP start points, one can expect to collect

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

DP chains of length j.

As we do not store the chain length information, the number of iterations required when a collision happens is $\hat{t}-i+1$. The only exception is when a pre-image is found, which is rare enough to be ignored.

Now referring to Proposition 5, the number of extra one-way function invocations induced by alarms can be expressed as

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

This can be approximated by the integration

$$\frac{\frac{mt^2}{N}t}{1-e^{-\hat{t}/t}} \int_0^{\hat{t}/t}\int_0^{\hat{t}/t} exp(-u)exp(-v)\{exp(min\{u,v\}-1)(\frac{\hat{t}}{t}-u)\}dvdu.$$

We reach the claimed value by replacing $\hat{t}/t$ by c and computing this definite integral. □

The following time memory tradeoff curve of DP tradeoff is obtained using the same approach as in [5].

Proposition 6. *The time memory tradeoff curve is $TM^2 = D_{tc}N^2$, where the tradeoff coefficient is*

$$D_{tc} = \{(-2+c+2ce^{-c}+e^{-c}+e^{-2c}-\frac{c^2}{2}e^{-c})\cdot D_{msc}+(1-e^{-c})^2\}\frac{D_{ps}\{ln(1-D_{ps})\}^2}{(1-e^{-c})D_{cr}^3 D_{msc}}. \tag{7}$$

Now we can find optimal parameter sets by minimizing the tradeoff coefficient and obtain Proposition 1. We drop from Equation 7 any part that depends only on D_{ps} and consider

$$D_{tmp}[D_{msc},c] = \frac{(-2+c+2ce^{-c}+e^{-c}+e^{-2c}-\frac{c^2}{2}e^{-c})\cdot D_{msc}+(1-e^{-c})^2}{(1-e^{-c})D_{cr}^3 D_{msc}}. \tag{8}$$

We substitute D_{cr} given by Proposition 4 into Equation 8. $D_{tmp}[D_{msc},c]$ can be viewed as a function of variables D_{msr} and c. We plot D_{tmp} and using numerical methods we find that the minimum value of $D_{tmp} = 6.5452$ is obtained at $D_{msc} = 1.47$ and $c = 1.97$. Referring to Proposition 4 again, we get the rest optimal parameters and this leads us to the claim of Proposition 1.