TMTO With Multiple Data: Analysis and New Single Table Trade-offs

Sourav Mukhopadhyay and Palash Sarkar
Cryptology Research Group
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
e-mail:{sourav_t,palash}@isical.ac.in

Abstract

Time/memory trade-off (TMTO) was introduced by Hellman and later studied by many other authors. The effect of multiple data in Hellman TMTO was studied by Biryukov and Shamir (BS). We continue the analysis of the general multiple data TMTO started in BS. The trade-offs of Babbage and Golic (BG) and Biryukov-Shamir are obtained as special cases. Further, the general analysis is carried out under different conditions including that of Hellman optimality (online time equal to memory). Our main contribution is to identify a new class of single table multiple data trade-offs which cannot be obtained either as BG or BS trade-off. In certain cases, these new trade-offs can provide more desirable parameters than the BG or the BS methods. We consider the analysis of the rainbow method of Oechslin and show that for multiple data, the TMTO curve of the rainbow method is inferior to the TMTO curve of the Hellman method. The costs of the rainbow method and the Hellman+DP method can be comparable if the size of the search space is small and the cost of one table look-up is relatively high.
Keywords: time/memory trade-off, one-way function.

1 Introduction

In 1980, Hellman [6] introduced the technique of time/memory trade-off (TMTO) attack on block ciphers. In its more general form, this can be visualised as a general one-way function inverter. See [5] for theoretical work on TMTO. The original work by Hellman considered inverting a one-way function f at a single data point.

TMTO appeared in the context of stream cipher in the works of Babbage [1] and Golic [4], jointly called the BG attack. The idea in these two attacks was to find the internal state of the stream cipher. These two papers considered the use of multiple data in TMTO, i.e., the inversion of f at any one of a set of points. In a later work, Biryukov and Shamir [2] combined the BG attack with the Hellman attack, to obtain a better multiple data TMTO.

We perform an analysis of Hellman attack in the presence of multiple data. The BG and the BS attacks are obtained as special cases of the general attack. So far, it has been believed that any multiple data TMTO is either BG or BS. Our analysis shows that this is not true. In particular, we show that the BS attack is good for multiple tables. On the other hand, the BG attack uses a single table and a single column. Our work reveals that there are other possible desirable trade-offs for *single table multiple column* attacks which are not obtainable from either the BG or the BS attack.

The general analysis is carried out under two different assumptions: online time equal to memory as considered originally by Hellman [6]; and equal online and offline times as briefly considered in [2]. Both these cases provide new single table trade-offs which were not known earlier. The time required for table look-up can be reduced by using Rivest's idea of distinguished point (DP) method. We analyze this method in a general setting. Finally, we consider the rainbow method and show that in the presence of multiple data, the TMTO curve of the rainbow method is inferior to the TMTO curve of the Hellman method. For the case of small size search space and relatively high cost of one table look-up, the online time of both the attacks can be comparable.

We organize this paper as follows: Section 2 presents the time/memory/data trade-off methodology. In Section 3, we analyze Hellman and DP methods in a general setting in the presence of multiple data. Section 4 provides the new single table trade-off (single table multiple column). We analyze the rainbow method in presence of multiple data in Section 5. To increase the success probability in Hellman method with $D = 1$, Kim and Matsumoto [7] increases the search space (total number of elements in the table which are not necessarily distinct) from N to λN for $\lambda > 1$. In Section 6, we apply the same technique in the presence of multiple data by considering the search space to be $\lambda \frac{N}{D}$ where $\lambda \geq 1$. Finally we conclude in Section 7.

2 Time/Memory/Data Trade-Off Methodology

The basic idea of Hellman's TMTO is quite simple. Suppose $f : \{0,1\}^n \rightarrow \{0,1\}^n$ be a one-way function. The algorithm consists of two stages: a one-time offline stage followed by an online stage. In the online stage, we will be given D points $y_1, \ldots, y_D$ in the range of f and we require to find the pre-image of any one of these points. In the offline stage, a set of tables are prepared covering N/D of the domain points, where $N = 2^n$. Since N/D domain points are covered and the online stage involves D domain points, by the birthday bound, we are assured of constant probability of success in finding the pre-image of one of the y's.

In the offline stage r tables are prepared. Let $f_1, \ldots, f_r$ be simple output modifications of f ($f_i = g_i \circ f$), such that the f_i's can be assumed to be pairwise independent. The ith table is prepared in the following manner. A total of m random domain points are chosen. For each domain point, the function f_i is iteratively applied t times to reach an end point. The pairs (start-point, end-point) are stored as part of the ith table in the sorted order of the end points. The total storage requirement is rm pairs of points, while the total coverage is rmt points.

In the online stage, for each data point y_j we look for a pre-image in the set of points covered by the tables. For searching in the ith table, we first apply g_i to y_j to obtain y'_j, then we iteratively apply f_i a total of t times to y'_j. After each application of f_i, we look in the end points of the

ith table for a match. If a match is found, we go to the corresponding start point and iterate f_i until we reach y'_j. The preceding point is a possible pre-image of y, which is verified by applying f to it and checking whether we obtain y. This requires a total of t applications of f and t table look-ups per table per data item. For more details regarding the method see the original paper by Hellman [6]. Also [2] provides a good description of the method.

We discuss some general issues about TMTO. In Hellman's original scheme, $D = 1$; the table preparation time is disregarded and only the online time and memory requirements are considered. The assumption is that the tables would be prepared once for all in an offline phase. Once the tables are prepared, they will not change and can be used to find different pre-images. In this scenario, the table preparation time can be huge and even larger than exhaustive search. Thus, the security of a cryptographic algorithm with respect to this kind of TMTO has a hidden cost of offline (and one time) exhaustive search.

If multiple data is available, the actual table preparation time will be less than exhaustive search. An extension of Hellman's original principle would be to disregard the table preparation irrespective of whether it is less or more than exhaustive search.

On the other hand, in the presence of multiple data, we might wish to consider the possibility of a feasible attack even taking table preparation time into account. Since this is an offline activity, it might be reasonable to expect the table preparation time to be more than the online time but less than exhaustive search time. One possibility (briefly considered in [2]) is to consider both offline and online time to be of equal importance and require that they are both equal. This can lead to interesting trade-offs as we consider later.

The amount of available data is certainly a lower bound on the online time. It can also be a lower bound on the memory requirement. This is because the data usually needs to be stored before processing. Another possible scenario is that the data is processed as it is received thus doing away with the necessity of storing it in which case the data can be more than the memory. This might happen, for example, in the BG attack.

The precomputation time will be in general more than the memory requirement. In the table preparation stage, the entire table will have to be computed and only a fraction of it stored. This shows that the offline time will be at least as large as the memory requirement.

Hellman in his original paper [6], considered the condition where the online time is equal to the memory requirement. This provides a kind of optimality condition, which we will call Hellman optimality. In the presence of multiple data, it is perhaps more practical to require the data and memory requirement to be less than the online and offline time requirements. This has been considered in [2]. We consider the situation where the data and memory requirement are taken to be equal and less than the online and offline times which are also taken to be equal.

The data requirement denotes the number of blocks of data. Each block is n bits and D is the data requirement, then it might appear that the total data requirement is nD bits. However, it can be substantially lower [1, 4]. For a bit oriented stream cipher, using a one-bit sliding window, D blocks might be obtainable from as low as $D + n - 1$ bits.

3 The Hellman Attack

Suppose r tables each of dimension $m \times t$ are used and the online data consists of D points. Then from the Hellman attack we have the following relations.

$$\left.\begin{array}{lcll} T_f & = & r(t-1)D & \text{(\# } f \text{ invocations in the online phase)} \\ T_t & = & rtD & \text{(\# table look-ups in the online phase)} \\ P & = & rmt & \text{(\# } f \text{ invocations in the pre-computation phase)} \\ & = & \frac{N}{D} & \text{(coverage)} \\ M & = & rm & \text{(memory)} \\ mt^2 & \leq & N & \text{(birthday bound)} \end{array}\right\} \tag{1}$$

If $t \gg 1$, we can assume $t-1 \approx t$ and $T_f \approx rtD = T_t$. We will usually make this assumption, except for the analysis of the BG attack, where $t = 1$. Let γ be the ratio of the time required for performing one table look-up to the time required for one invocation of f, i.e.,

$$\gamma = \frac{\text{time for one table look-up}}{\text{time for one invocation of } f}. \tag{2}$$

We assume that one unit of time corresponds to one invocation of f (i.e., one unit of time = time required for completing one invocation of f). The time required for the table look-ups is then γrtD. Define $T = \max(T_f, \gamma T_t) = \gamma rtD$. The parameter T is a measure of the time required during the online phase. The actual online time is proportional to $T_f + \gamma T_t$. However, this is only at most twice the value of T. Thus, we will perform the analysis with T instead of $T_f + \gamma T_t$.

For the present, we will assume that $\gamma = 1$ (and $T = T_t \approx T_f$), i.e., the cost of one invocation of f is equal to the cost of one table look-up. The value of γ need not actually be one; even if it is a small constant (or a negligible fraction of N), we can assume it to be one and that will not affect the asymptotic analysis. On the other hand, [2] mentions that γ may be as large as one million ($\approx 2^{20}$). If N is only moderately large (like 2^{64} for A5/1), then γ can be a significant proportion of N. In such a situation, we cannot assume $\gamma = 1$ and the cost of table look-up will dominate the total online cost. This case will be considered later.

Using (1), we can solve for r, m and t as follows.

$$\left.\begin{array}{lcll} t & = & \frac{N}{MD} \geq 1 & \text{(number of columns)} \\ m & = & \frac{N}{T} & \text{(number of rows)} \\ r & = & \frac{MT}{N} \geq 1 & \text{(number of tables)} \\ mt^2 & = & \frac{N^3}{TM^2D^2} \leq N & \text{(birthday bound)} \end{array}\right\} \tag{3}$$

Note that all three of r, m and t must be at least 1. Since $m = N/T$ and for a valid attack we must have $N > T$, the condition on m is trivially satisfied. The advantage of writing in the form of (3) is that given values for T, M and D satisfying the proper constraints, we can immediately design a table structure which achieves these values.

Let $D = N^a$ for some $0 \leq a < 1$. Since $PD = N$, we have $P = N^{1-a}$. The condition on r, shows that $MT \geq N$. We write $MT = N^b$ for $b \geq 1$. Also let $M = N^c$; for a valid attack we must have $0 \leq c < 1$. Since $MT = N^b$, we have $T = N^{b-c}$ and again for a valid attack, we must have $0 \leq b - c < 1$. The available online data D is a lower bound on M and T and hence we have $a \leq c, b - c$. Since the birthday bound tells us that $mt^2 \leq N$, we write $mt^2 = N^d$, for some d with $0 \leq d \leq 1$. Substituting in the last equation of (3), we obtain $2a + b + c + d = 3$. The condition on t shows that $MD \leq N$, which translates to $a + c \leq 1$. Thus, any set of values for a, b, c and d which satisfy the following constraints constitute a valid attack.

$$\left.\begin{array}{ll} \textbf{C1:} & 2a + b + c + d = 3 \\ \textbf{C2:} & 0 \leq a < 1 \\ \textbf{C3:} & 0 \leq c, b - c < 1 \leq b \\ \textbf{C4:} & a + c \leq 1 \\ \textbf{C5:} & 0 \leq d \leq 1 \end{array}\right\} \tag{4}$$

The so-called TMTO curve can be obtained as the following relations.

$$\left.\begin{array}{lcl} TM^2D^2 & = & N^{3-d} \\ PD & = & N \\ MD \leq & N & \leq MT \\ M, D, T & < & N. \end{array}\right\} \tag{5}$$

Also, we have the following values of r, m and t.

$$r = N^{b-1}; \quad m = N^{1-(b-c)}; \quad t = N^{1-a-c}. \tag{6}$$

Since $MT = N^b \geq N$, we have $r = 1$ if and only if $MT = N$. With $r = 1$, we have only one table and hence if there are more than one tables, then MT is strictly greater than N.

BG Attack [1, 4]: In this case, we have $r = t = 1$. This implies $T_f = 0$, i.e., the online phase does not require invocation of f. The cost in the online phase is $T = T_t$ and we have $MD = N = MT$ and hence $T = D$; $M = N/D$. This corresponds to the conditions $a + c = 1; b = 1; d = 1 - a$.

BS Attack [2]: In [2], $r = t/D$ and $d = 1$ is used. Then $T = t^2$, $M = mt/D$ and hence $r = N^{-a+(b-c)/2}$. Since $r \geq 1$, we have the restriction $0 \leq 2a \leq b - c$ (i.e., $1 \leq D^2 \leq T$) in addition to (4).

The conditions $d = 1$ and $r = t/D$ are related (e.g., if $r = 1$, then $t = D$ and $T = t^2 = D^2$). In the following analysis, we will proceed without these two conditions. Later, we show the situation under which making these two assumptions is useful.

3.1 Condition $P = T$

Since both P and T represent time, the case $P = T$ puts equal emphasis on both the offline and the online times. The condition $P = T$ implies $P = N^{1-a} = T = N^{b-c}$ and so $m = N^{1-(b-c)} = N^a = D$. (On the other hand, $P = M$ is possible only if $t = 1$.) Since $PD = N$, we have $T = N/D$ and so the curve becomes $M^2D = N^{2-d}$. If $P = T$, then $r = M/D$. If further $M = D$, then $M = D = N^{(2-d)/3}$ and $P = T = N^{(1+d)/3}$.

Proposition 1 *If $P = T$ and $M = D$ in (5), then $M = D = N^{(2-d)/3}$ and $P = T = N^{(1+d)/3}$. Further, $r = 1$, i.e., exactly one table is required.*

Proposition 1 gives us a nice way to control the trade-off between time and data/memory requirement by varying d. Choosing $d = 1$, corresponds to $(P, D, M, T) = (N^{2/3}, N^{1/3}, N^{1/3}, N^{2/3})$ and has been observed in [2]; choosing $d = 1/2$ corresponds to $(P, D, M, T) = (N^{1/2}, N^{1/2}, N^{1/2}, N^{1/2})$ which is the square root birthday (BG) attack.

3.2 Condition $T = M$

The condition $T = M$ was considered by Hellman [6] to be an optimality condition. We perform an analysis of (5) with $T = M$. Then $c = b - c$, whence $c = b/2$. Condition **C1** becomes $2a + 3c + d = 3$ and so

$$\frac{b}{2} = c = 1 - \frac{2a + d}{3}. \tag{7}$$

Using $a + c \leq 1$, we obtain $a \leq d$. Also since $b \geq 1$, we have $c = b/2 \geq 1/2$. This along with (7) gives $d \leq 3/2 - 2a$. Since, we already know $d \leq 1$, we obtain

$$a \leq d \leq \min(1, \frac{3}{2} - 2a). \tag{8}$$

Thus, any non-negative solution in a and d to (8) gives a valid attack with $T = M = N^c$.

We are interested in minimizing the value of c. From (7), we see that the value of c is minimized by maximizing the value of d. In fact, using (8), we can choose $d = 1$ as long as $1 \leq \frac{3}{2} - 2a$, i.e., $2 - (1/2a) \leq 0$ or $a \leq 1/4$. Thus, for $a \leq 1/4$, we obtain $T = M = N^{b/2} = N^{(2-2a)/3}$.

In the case $3/2 - 2a \leq 1$, we have $a \leq d \leq 3/2 - 2a$. For the gap to be non-empty we must have $a \leq 1/2$. For minimizing c, we use the upper bound, i.e., $d = 3/2 - 2a \leq 1$. Thus, for $1/4 \leq a \leq 1/2$, we have $c = 1/2$ and $T = M = N^{1/2}$. Finally, we obtain the following result.

Theorem 2 *If $T = M$, then $D \leq N^{1/2}$ and the following conditions hold.*

1. $N^{1/2} \leq T = M = N^{(2-2a)/3} \leq N^{2/3}$, *for* $1/4 \geq a \geq 0$.
2. $T = M = N^{1/2}$, *for* $1/4 \leq a \leq 1/2$.

For the first case we have, $(a, b, c, d) = (a, 2(2-2a)/3, (2-2a)/3, 1)$ and for the second case we have $(a, b, c, d) = (a, 1, 1/2, 3/2 - 2a)$. The corresponding values of (r, m, t) are $(N^{(1-4a)/3}, N^{(1+2a)/3}, N^{(1-a)/3})$ and $(1, N^{1/2}, N^{1/2-a})$ respectively.

In the second case of Theorem 2, exactly one table is required. However, it is not the BG attack, since the number of columns can be more than one. Also, we have $T \leq P \leq N$. The situation with $T < P < N$ is interesting, since the pre-computation time is less than exhaustive search. Even though P is more than T, since it is an offline activity, we might wish to spend more time in the pre-computation part than in the online attack.

3.3 Distinguished Point Method

We now consider the case where $\gamma \gg 1$. In this case, a direct application of the Hellman method leads to $T = \gamma rtD$, i.e., the time required for the table look-ups dominate the online time. It is useful to consider the distinguised point method of Rivest to reduce the number of table look-ups. See [2] for a description of the DP method.

Using the distinguished point method results in reducing the number of table look-ups from rtD to rD, i.e., one table look-up per table per data. Then $T_t = rD = N^{a+b-1}$. (Note $T_t = N^a = D$, i.e., only one table look-up is required per data item if and only if $b = 1 = r$, i.e., $MT = N$.)

The total cost of the table look-ups is γrD whereas the cost of invoking the one-way function is rtD. In this case, the ratio of the two costs is γ/t. If $t \geq \gamma$, then the ratio is at most one. Hence, we can again ignore the cost of table look-up and perform the analysis by considering simply the cost of invoking the one-way function. The actual runtime will be at most twice the runtime obtained by such an analysis.

Suppose $t < \gamma$. Then the analysis performed above does not hold. We now investigate the situation under which $t < \gamma$ holds. This certainly holds for $t = 1$ (the BG attack), but in the BG attack the entire online computation consists of table look-ups and hence the general analysis is not required. Recall that $t = N^{1-(a+c)} = 2^{n(1-(a+c))}$, $D = N^a$ and $M = N^c$. Suppose $\gamma = 2^e$. Then $t \geq \gamma$ if and only if $a + c \leq 1 - (e/n)$. The value of e is a constant whereas n increases. Hence, $(1 - e/n) \to 1$ as n grows. Thus, we can have $a + c > 1 - e/n$ only for small values of n. The smallest value of n for which we can expect to have a secure cryptographic algorithm is 64. Further, as mentioned in [2], e can be at most around 20 and so $1 - e/n \geq 2/3$ for $n \geq 64$.

Consider $a = c = 1/3$, as in the solution $(a, b, c, d) = (1/3, 1, 1/3, 1)$ corresponding to $P = T = N^{2/3}$; $M = D = N^{1/3}$; $r = 1$ of [2]. If $n \geq 64$, then $a + c = 2/3 \leq 1 - e/n$ and the time analysis assuming $T = rtD = tD$ holds. On the other hand, for the solution $(a, b, c, d) = (3/8, 1, 3/8, 7/8)$ corresponding to $P = T = N^{5/8}$; $M = D = N^{3/8}$; $r = 1$ considered in Section 4, we have $a + c = 3/4$. For $n = 64$, $a + c > 1 - e/n$ and we have to assume $T = \gamma rD = \gamma D$, whereas for $n = 100$, $a + c \leq 1 - e/n$ and we can assume $T = rtD = tD$. Thus, for relatively small n, we should solve (4) with the constraint $a + c \leq 1 - e/n$ instead of $a + c \leq 1$. This disallows some of the otherwise possible trade-offs.

There is another issue that needs to be considered. We have to ensure that t is large enough to ensure the occurrence of a DP in a chain. Let 2^{-p} be the probability of a point being a DP. Hence, we can expect one DP in a random collection of 2^p points. Thus, if $t \geq 2^p$, we can expect a DP in a chain of length t. This implies $p \leq \log_2 t$. Any attempt to design the tables with $t < 2^p$, will mean that several trials will be required to obtain a chain terminating in a DP. This will increase the pre-computation time. In fact, [2] has shown that use of the DP method in the BG attack divides into two different trade-offs leading to unrealistic requirements on data and memory.

Using (6), we have $\frac{p}{n} \leq 1 - (a + c)$. This leads to the condition $a + c \leq 1 - \frac{p}{n}$ ($MD \leq N^{1-\frac{p}{n}}$) instead of the condition $a+c \leq 1$ (resp. $MD \leq N$) in (4) (resp. (5)). For small n, this condition has to be combined with $a+c \leq 1-e/n$ and we should solve (4) with the constraint $a+c \leq 1-\frac{1}{n}\max(p, e)$ instead of the constraint $a + c \leq 1$. This puts further restrictions on otherwise allowed trade-offs.

3.3.1 BSW Sampling

There is an elegant application of TMTO in [3], which uses a special type of sampling technique called the BSW sampling. This technique uses only part of the available online data and also reduces the search space. The trade-off curve does not change, but the number of table look-ups reduces significantly. Use of this technique allowed particularly efficient attacks on A5/1.

Use of the BSW technique reduces the amount of available online data. This makes it difficult to use a single table to carry out the TMTO. In such a situation, our analysis does not lead to any new insight into the BSW technique. On the other hand, if the available online data (even after sampling) is large enough to allow the use of a single table, then our analysis applies and one can consider a wider variety of trade-offs.

4 Single Table Attack

The case $N = 2^{100}$ has been considered in [2]. It has been mentioned in [2] that the Hellman attack with $D = 1$; $T = M = N^{2/3} = 2^{66}$ requires unrealistic amount of disk space and the BG attack with $T = D = N^{2/3} = 2^{66}$; $M = N^{1/3} = 2^{33}$ requires unrealistic amount of data. (Note $T = M = D = N^{1/2} = 2^{50}$ also gives a BG attack. However, as mentioned in [2] in a different context, data and memory requirement of more than 2^{40} is unrealistic.) Further, [2] mentions $P = T = 2^{66}$ and $D = M = 2^{33}$ to be a (barely) feasible attack. This corresponds to the parameters $(a, b, c, d) = (1/3, 1, 1/3, 1)$ and $(r, m, t) = (1, N^{1/3}, N^{1/3})$.

From Proposition 1, if we choose $d = 7/8$, then we obtain $M = D = N^{3/8} = 2^{37.5}$ and $P = T = N^{5/8} = 2^{62.5}$. The corresponding parameters are $(a, b, c, d) = (3/8, 1, 3/8, 7/8)$ and $(r, m, t) = (1, N^{3/8}, N^{1/4})$. This brings down the attack time while keeping the data and memory within feasible limits. Since $t > 1$, this cannot be obtained from the BG attack. Further, choosing $d = 7/8$ and $D^2 > T$ ensures that this attack cannot also be obtained from the BS attack. We would like to point out that [2] mentions that choosing $d < 1$ is "wasteful". The above example shows that this is not necessarily the case and choosing $d < 1$ can lead to more flexible trade-offs. We show below the condition under which choosing $d < 1$ is indeed "wasteful".

The choice of the parameter $r = t/D$ is motivated in [2] by mentioning that this reduces the number of table look-ups. The number of table look-ups in the first case is $rtD = tD = N^{2/3}$ whereas in the second case, it is $rtD = tD = N^{5/8}$. Thus, the above example shows that the condition $r = t/D$ is not necessary for reducing the number of table look-ups.

As mentioned earlier, we have one table (i.e., $r = 1$) if and only if $MT = N$. The reason for moving to more than one tables is when $mt^2 > N$ and we begin to have more and more repetitions within a table.

Proposition 3 *There is a solution to (5) with $r = 1 = b$ (and hence $MT = N = PD$) if and only if $2a + c \geq 1$.*

Proof : Suppose $r = 1$. Then $b = 1$ and $2a + c + d = 2$. Hence $d = 2 - (2a + c)$. Since $d \leq 1$, this shows $2a + c \geq 1$.

On the other hand assume that $2a + c \geq 1$. Choose $b = 1$ and set $d = 2 - (2a + c) \leq 1$. This choice satisfies the conditions of (5). Further, since $b = 1$, we have $r = 1$. ∎

Suppose $2a+c<1$. Then $b+d>2$ and $b>2-d$. Since $MT=N^b$, we would like to minimize b and hence we choose $d=1$. We can now modify the suggestion of [2] and say that it is "wasteful" to choose $mt^2<N$ *if there are more than one table.* Since $b>1$, we have $2a+c<1<b$ and hence $2a<b-c$ which gives $D^2<T$ and we are back to the situation described in [2].

Thus, the analysis of [2] actually applies to the situation where the data is small enough to require more than one tables. On the other hand, for the case of one table, the restrictions of [2] are not required and removing these restrictions provide more flexible trade-offs. We would like to point out that there are interesting situations where a single table can be used. Apart from the examples $D=M=N^{1/3}$ and $D=M=N^{3/8}$ already considered, other possible examples are $(D=N^{0.3}, M=N^{0.4})$; $(D=N^{0.25}, M=N^{0.5})$, etcetra.

Going back to the example of $N=2^{100}$, both $(P,D,M,T)=(N^{2/3},N^{1/3},N^{1/3},N^{2/3})$ of [2] and $(P,D,M,T)=(N^{5/8},N^{3/8},N^{3/8},N^{5/8})$ described above have $r=1$. As mentioned above, the second one is better with respect to the number of table look-ups. In conclusion, there are reasonable choices of data and memory requirements which lead to a single table. In such situations, the trade-off in [2] is not the only possible one. Other (and perhaps better) trade-offs can be obtained following the approach described here. We highlight some of the other interesting single table trade-offs that can be obtained.

Condition $P=T=N^{(1+d)/3}$; $M=D=N^{(2-d)/3}$: From Proposition 1, we have $r=1$, i.e., all trade-offs attaining this condition use a single table. In the plausible situation, $M=D\leq P=T$, we have $1/2\leq d\leq 1$. The case $d=1$ can be obtained from the BS analysis. In the BG analysis, we have $d=1-a$. Since $a-(2-d)/3$, this condition leads to $d=1/2$. Thus, the range $1/2<d<1$ for which the condition $P=T=N^{(1+d)/3}$; $M=D=N^{(2-d)/3}$ can be attained was not known earlier.

Condition $M=T$: In the second case of Theorem 2, we have $r=1$ and $M=T=N^{1/2}$. The allowed range of a for this case is $1/4\leq a\leq 1/2$. The case $a=1/4$ can be obtained from the BS analysis and the case $a=1/2$ can be obtained from the BG analysis. However, the range $1/4<a<1/2$ for which $T=M=N^{1/2}$ can be attained, cannot be obtained from either the BG or the BS analysis and provide previously unknown trade-offs. The advantage is that the data can be increased (thus lowering offline time) without increasing either time or memory.

Small N Consider $N=2^{64}$, as in A5/1. It is mentioned in [3] that $M\approx 2^{35}$ and $D\approx 2^{22}$ are reasonable choices. We consider two trade-offs, corresponding to the second case of Theorem 2.

Trade-Off 1: $(P,D,M,T)=(2^{46},2^{18},2^{32},2^{32})$: The table parameters are $(r,m,t)=(1,2^{32},2^{14})$.

Trade-Off 2: $(P,D,M,T)=(2^{42},2^{22},2^{32},2^{32})$: The table parameters are $(r,m,t)=(1,2^{32},2^{10})$.

None of the above two trade-offs are obtainable as BG trade-offs, since in both cases $t>1$. Also, neither can be considered to be BS trade-offs since $D^2>T$. For both trade-offs, the data and memory are within reasonable limits and the online times are the same. The offline time is lower for the second trade-off and is within doable limits (especially as an offline one-time activity), while for the first attack it is probably just outside the doable limit.

The total cost of online table look-up for both the attack is $\gamma t D$. Since the value of γ is a significant proportion of N the cost of table look-up dominates the online cost. Use of the DP method reduces the total cost of table look-ups to γD. If γ is around 2^{20} as mentioned in [2], we have the table look-up costs to be 2^{38} and 2^{42} respectively. This pushes up the online cost of both the attacks to make them less of a threat. On the other hand, if γ can be brought down to around 2^{10} by the deployment of special purpose high speed memory, then the table look-up costs come down to 2^{28} and 2^{32} respectively. This will make both the attacks serious threats. We note that with $\gamma \approx 2^{20}$, the attack of [3] remains the most efficient one.

5 The Rainbow Attack

The rainbow attack was introduced in [8]. The number of table look-ups of the rainbow method is comparable to that of the Hellman+DP method. See [8] for a discussion of the relative advantages of the rainbow method with respect to the DP method.

In the rainbow attack, we use a table of size $m \times t$ and suppose there are D online data points. Then the total number of invocations of the one-way function is $t^2D/2$ while the cost of the table look-ups is tD. Again, we will ignore the factor of two in the runtime since it does not significantly affect the analysis. Then, the total number of invocations of f is t^2D and the total number of table look-ups is tD. Also, we have $mt = N/D$.

If we assume $\gamma \approx 1$, then the cost of invoking f dominates the online cost and we have $M = m$ and $T = t^2D$. Assume $D = N^a$ and $M = N^c$ as in the case of Hellman analysis. Then since $mt = N/D = N^{1-a}$, we have $t = N^{1-a-c}$ and $T = t^2D = N^{2-a-2c}$. Also, since $t \geq 1$, we must have $a + c \leq 1$. The TMTO curve for rainbow in the presence of multiple data is $TM^2D = N^2$ which is inferior to the Hellman TMTO curve when $D > 1$.

We now compare the rainbow parameters $(P, D, M, T) = (N^{1-a}, N^a, N^c, N^{2-a-2c})$ with the Hellman parameters for same data and memory. For multiple table Hellman, we choose $d = 1$ and hence the corresponding Hellman parameters are $(P, D, M, T) = (N^{1-a}, N^a, N^c, N^{2-2a-2c})$. If $a > 0$, i.e., if multiple data is available, then clearly Hellman time is less than rainbow time.

If γ is a significant fraction of N, then the cost of table look-ups is $\gamma t D$ while the cost of invoking f is still t^2D. In the case $\gamma > t$, which happens for relatively small N (around 2^{64} or so), the cost of table look-up dominates the online cost. To compare to the Hellman method we have to consider the Hellman+DP algorithm. For the case $\gamma > t$, the online cost of the Hellman method is also $\gamma t D$. Hence, for this case, the costs of online time for the rainbow and the Hellman+DP methods are equal. In this situation, one might prefer to use the rainbow method for the possibly lower rate of false alarms compared to the DP method [8].

Thus, we conclude that in the presence of multiple data, in general the Hellman attack is better than the rainbow attack. For the case of small N, the online times of both attacks can be comparable and one might prefer rainbow for obtaining other possible advantages.

6 Increasing Coverage Space

The success probability of the Hellman method is constant. It has been observed [7] that this value is around 60%. To increase the success probability, one can increase the coverage space of all the tables. The tables together cover a total of rmt points. We assume that $rmt = \lambda(N/D)$ for some $\lambda \geq 1$. By choosing $\lambda > 1$, it is possible to increase the success probability. Kim and Matsumoto [7], have described this technique for the basic Hellman attack with $D = 1$. Below we show that essentially the same technique also works for $D > 1$.

Let $y_1, \ldots, y_D$ be the D data points and let $x_1, \ldots, x_D$ be such that $f(x_i) = y_i$. We define $\mathsf{P}_{\mathsf{Succ}}$, the success probability to be the probability that at least one of x_i is in the tables. Let P_1 be the probability that a random point is covered by the tables and E_i be the event that x_i is not in the tables. Then $\mathsf{Prob}(E_i) = 1 - P_1$ for $1 \leq i \leq D$.

$$\begin{aligned}
\mathsf{P}_{\mathsf{Succ}} &= 1 - \mathsf{Prob}(E_1 \cap E_2 \cap \ldots \cap E_D) \\
&= 1 - \prod_{i=1}^{D} \mathsf{Prob}(E_i) \\
&= 1 - (1 - P_1)^D \qquad (9)
\end{aligned}$$

To find P_1, we proceed as follows. Let $\mathsf{PS}_{\mathsf{single}}$ be the probability that the randomly chosen point is in a single table and A_j be the event that the point is not in the j^{th} table. Then, $\mathsf{Prob}(A_j) = 1 - \mathsf{PS}_{\mathsf{single}}$.

$$\begin{aligned}
P_1 &= 1 - \mathsf{Prob}(A_1 \cap A_2 \cap \ldots \cap A_r) \\
&= 1 - \prod_{i=1}^{r} \mathsf{Prob}(A_i) \\
&= 1 - (1 - \mathsf{PS}_{\mathsf{single}})^r \qquad (10)
\end{aligned}$$

Hence,

$$\mathsf{P}_{\mathsf{Succ}} = 1 - (1 - P_1)^D = 1 - (1 - \mathsf{PS}_{\mathsf{single}})^{rD}.$$

In [6], Hellman provided the expression

$$\mathsf{PS}_{\mathsf{single}} \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left(1 - \frac{it}{N}\right)^j.$$

Later Kim and Matsumoto [7] made the following simplification:

$$\mathsf{PS}_{\mathsf{single}} \geq \frac{1}{N} \sum_{i=1}^{m} \sum_{j=1}^{t} \left(1 - \frac{it}{N}\right)^j \approx \frac{1}{t} \sum_{i=1}^{m} \frac{1 - e^{\frac{-it^2}{N}}}{\frac{it}{N}} \frac{t}{N} \approx \frac{1}{t} \int_0^{\frac{mt}{N}} \frac{1 - e^{-tx}}{x} dx \approx h(u) \frac{mt}{N},$$

where $h(u) = \frac{1}{u} \int_0^u \frac{1-e^{-x}}{x} dx$ and $u = \frac{mt^2}{N}$. This gives

$$\mathsf{P}_{\mathsf{Succ}} = 1 - (1 - \mathsf{PS}_{\mathsf{single}})^{rD} \geq 1 - \left(1 - h(u)\frac{mt}{N}\right)^{rD} \approx 1 - \mathsf{e}^{(-h(u)\frac{rmtD}{N})} = 1 - \mathsf{e}^{(-h(u) \times \lambda)},$$

where $\lambda = \frac{rmtD}{N}$. Now $h(u) = \frac{1}{u}\int_0^u \frac{1-e^{-x}}{x}dx < 1$ for all $u > 0$. This implies that the success probability increases with the value of λ.

Thus, we should first fix λ to achieve the desired success probability. Since λ usually turns out to be a small constant, its effect on P, T, M and D is negligible. Hence, the analysis carried out above will hold, with the actual attack parameters increasing by at most a constant factor.

7 Conclusion

We have studied the general problem of utilising multiple data in time/memory trade-off attacks introduced by Hellman in [6]. We build on the analysis performed by Biryukov and Shamir [2]. Our general analysis shows that the Babbage-Golic attack and the Biryukov-Shamir attacks are special cases of the general Hellman attack. We also analyse under the Hellman optimality condition ($T = M$) and under the assumption $P = T$. Our main new contribution is the identification of a new class of single table trade-offs which are not obtainable as either the BG or the BS attacks. In certain cases, these new trade-offs can be more desirable than the previous BG or the BS attacks. Finally, we consider the rainbow attack of Oechslin and show that with the utilization of multiple data, the TMTO curve of the rainbow attack is inferior to the TMTO curve of the Hellman attack. On the other hand, for small N and relatively high cost of one table look-up, the online time for both the rainbow and the Hellman+DP methods are comparable.

References

[1] S. Babbage. A Space/Time Tradeoff in Exhaustive Search Attacks on Stream Ciphers, European Convention on Security and Detection, IEE Conference Publication No. 408, May 1995.

[2] A. Biryukov and A. Shamir. Cyptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers, in the proceedings of ASIACRYPT 2000, LNCS, vol 1976, pp 1-13, 2000.

[3] A. Biryukov, A. Shamir and D. Wagner. Real Time Cryptanalysis of A5/1 on a PC, Proceedings of Fast Software Encryption 2000.

[4] J. Golic. Cryptanalysis of Alleged A5 Stream Cipher, Proceedings of Eurocrypt'97, LNCS 1233, pp. 239-255, Springer-Verlag 1997.

[5] A. Fiat and M. Naor. Rigorous time/space tradeoffs for inverting functions, In STOC 1991, pp 534-541, 1991.

[6] M. Hellman. A cryptanalytic Time-Memory Trade-off, IEEE Transactions on Information Theory, vol 26, pp 401-406, 1980.

[7] I.J. Kim and T. Matsumoto Achieving Higher Success Probability in Time-Memory Trade-Off Cryptanalysis without Increasing Memory Size, TIEICE: IEICE Transactions on Communications/Electronics/Information and System, pp 123-129, 1999.

[8] P. Oechslin. Making a faster Cryptanalytic Time-Memory Trade-Off, in the proceedings of CRYPTO 2003, LNCS, vol 2729, pp 617-630, 2003.