Applying Time-Memory-Data Trade-Off to Meet-in-the-Middle Attack

Jiali Choy, Khoongming Khoo, and Chuan-Wen Loe

DSO National Laboratories
20 Science Park Drive, Singapore 118230
Email: cjiali,kkhoongm,lchuanwe@dso.org.sg

Abstract. In this paper, we present several new attacks on multiple encryption block ciphers based on the meet-in-the-middle attack. In the first attack (GDD-MTM), we guess a certain number of secret key bits and apply the meet-in-the-middle attack on multiple ciphertexts. The second attack (TMTO-MTM) is derived from applying the time-memory trade-off attack to the meet-in-the-middle attack on a single ciphertext. We may also use rainbow chains in the table construction to get the Rainbow-MTM attack. The fourth attack (BS-MTM) is defined by combining the time-memory-data trade-off attack proposed by Biryukov and Shamir to the meet-in-the-middle attack on multiple ciphertexts. Lastly, for the final attack (TMD-MTM), we apply the TMTO-Data curve, which demonstrates the general methodology for multiple data trade-offs, to the meet-in-the-middle attack. GDD-MTM requires no pre-processing, but the attack complexity is high while memory requirement is low. In the last four attacks, pre-processing is required but we can achieve lower (faster) online attack complexity at the expense of more memory in comparison with the GDD-MTM attack. To illustrate how the attacks may be used, we applied them in the cryptanalysis of triple DES. In particular, for the BS-MTM attack, we managed to achieve pre-computation and data complexity which are much lower while maintaining almost the same memory and online attack complexity, as compared to a time-memory-data trade-off attack by Biryukov et al. at SAC 2005. In all, our new methodologies offer viable alternatives and provide more flexibility in achieving time-memory-data trade-offs.

Keywords. block cipher, meet-in-the-middle, time-memory-data trade-off, triple DES.

1 Introduction

In [4], Diffie and Hellman described a meet-in-the-middle (MTM) attack that can be applied on a multiple encryption cipher consisting of a concatenation of $L \geq 2$ block ciphers, each with independent keys. Due to this attack, the effective security of triple DES with a 168-bit key is reduced to 112 bits. We give a brief description of the MTM attack in Section 3.

There were subsequent improvements on the MTM attack after its discovery. One such improvement, described in [6, Section 7.37], is to apply a guess-and-determine technique to the MTM attack. We shall call this attack a guess-and-determine MTM attack (GD-MTM). The GD-MTM attack stipulates that the attacker must initially guess a certain number of key bits of the first block cipher. We propose an extension of this attack, which we call guess-and-determine MTM attack with multiple data (GDD-MTM) to cater for the case when a plaintext is encrypted by the cipher under several keys of which only one needs to be recovered. When applied to triple DES, as compared to the typical MTM attack without pre-computation which needs time complexity of 2^{112} and memory complexity of 2^{56}, the GD-MTM attack achieves lower memory complexity of 2^{48} at the cost of higher time complexity of 2^{120}, while the GDD-MTM attack achieves lower time complexity of 2^{104} at the expense of 2^{14} data and 2^{64} memory requirements. We expound on the GD-MTM and GDD-MTM attacks in Section 4.

In [5], Hellman introduced the time-memory trade-off (TMTO) attack which is a generic cryptanalytic attack that can be applied to any cipher algorithm using one data point. This attack circumvents exhaustive search by pre-computing and pre-storing a large amount of data. During the online phase, the attack uses the data generated by an unknown key to conduct a ciphertext or keystream comparison in order to recover the key quickly. The advantage of this attack is that with pre-computation done offline, the time taken in the online stage is shortened at the expense of more memory required. Later in 2003, Oechslin [8] proposed an improved technique called rainbow chains to derive a single pre-computation table. His method reduces the TMTO attack complexity by a factor of 2.

Then in [3], Biryukov et al. suggested a time-memory-data trade-off attack adapted from [2], which we shall call the BS attack. In the BS attack, they only need to find one key out of a family of ciphertexts which are the encryptions of the same plaintext under different keys. In this way, only a portion of the key space needs to be covered in the pre-computation phase. In addition, they also performed a new unified analysis of Hellman's attack in the presence of multiple data, thereby achieving new time-memory-data trade-offs previously unknown. This paved the way for more flexibility in attack modes depending on case-by-case time-memory-data requirements. The BS attack is a special case of this new framework which we name as the TMD attack.

However, for block ciphers with a large key size such as triple DES, we found that by applying the attacks suggested in [3], the data requirements and pre-computation complexity tend to be high. For example, for the BS attack on triple DES, the data complexity is 2^{42} and the pre-computation complexity is 2^{126}, with 2^{84} complexity for both time and memory. In Sections 5 to 7 of our paper, we address this issue by proposing novel ways of applying the TMTO, Rainbow, BS, and TMD attacks to the MTM attack. We derive four new attacks: the TMTO-MTM, Rainbow-MTM, BS-MTM, and TMD-MTM attacks. While the earlier two sections (on GD-MTM and GDD-MTM attacks) deal with attacks that involve no pre-computation, the last four attacks all require pre-computation.

When applying the TMTO-MTM attack to triple DES using single data, we found that the attack complexity was 2^{80} and the memory complexity was 2^{101}. The pre-processing stage has a complexity of 2^{113}. The Rainbow-MTM attack achieves a slightly better attack complexity of 2^{79} while maintaining the same requirements for memory and data. These can be considered as attacks with attack and memory complexities which are in between the MTM attack without precomputation (2^{112} attack complexity and 2^{56} memory complexity) and the MTM attack with precomputation (2^{56} attack complexity, 2^{113} pre-computation and memory complexities).

The BS-MTM attack is a further generalization of the TMTO-MTM attack in which the goal is now to recover one key out of several keys which are used to encrypt the same plaintext. This attack can be viewed as an improvement of the TMTO-MTM attack as the pre-computation complexity is reduced by a factor equal to the data complexity. Also, if the data, time and memory complexities are denoted by D, M and T respectively, then M^2T is reduced by a factor equal to at least D^2. When applied to triple DES, with pre-computation of 2^{99} and a data complexity of 2^{14}, we can achieve 2^{84} complexity for online attack time and 2^{85} complexity for memory. Thus it has comparable time and memory requirements as Biryukov et al.'s attack [3] but requires less data and pre-computation complexity. This will make the attack more feasible as it is usually very hard to obtain many encryptions (2^{42} compared to 2^{14}) of the same plaintexts under different keys. Moreover, a pre-computation complexity of 2^{99} seems more achievable than a pre-computation complexity of 2^{126}.

More flexibility is achieved by the TMD-MTM attack where we apply the new TMD trade-off in [3] to the MTM attack. For example, if we can afford a larger memory in our previous attack, e.g. 2^{99}, then we can reduce the online attack complexity to 2^{71} while keeping the data complexity at 2^{14} and pre-computation at 2^{99}.

2 Number of Plaintext-Ciphertext Pairs Needed for Verification

Suppose that the B-bit encryption function, E, is the composition of two block ciphers, $E1_{K_1}(\cdot)$ and $E2_{K_2}(\cdot)$, so that E acts on a plaintext P and outputs the ciphertext C given by

$$C = E_{(K_1,K_2)}(P) = E2_{K_2}(E1_{K_1}(P)).$$

Suppose that $K_1 \in GF(2)^{n_1}$, $K_2 \in GF(2)^{n_2}$, and that K_1, K_2 are independent. Then there will be at least $\lfloor \frac{n_1+n_2}{B} \rfloor$ keys mapping a given plaintext P to a certain ciphertext C. Therefore, in order to verify if a possible key (S_1, S_2) is indeed the cipherkey (K_1, K_2), $q = \lceil \frac{n_1+n_2}{B} \rceil$ plaintext-ciphertext pairs

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

need to be tested to check if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for all i. If so, then $(K_1, K_2) = (S_1, S_2)$.

In the case of multiple data where we only need to find one key out of several keys used to encrypt the same plaintext, the size of the key space is still fixed at $2^{n_1+n_2}$. Therefore, for both cases of single and multiple data, $q = \lceil \frac{n_1+n_2}{B} \rceil$ plaintext-ciphertext pairs are required to verify the correct key.

3 Background: Meet-in-the-Middle Attack (MTM)

The meet-in-the-middle attack is a well-known attack (e.g. see [6, page 235]). We shall give a description of it here. Suppose, as before, that the B-bit encryption function is the composition of two block ciphers so that

$$C = E_{(K_1,K_2)}(P) = E2_{K_2}(E1_{K_1}(P)),$$

where $K_1 \in GF(2)^{n_1}$, $K_2 \in GF(2)^{n_2}$. Assume further, that there are q plaintext-ciphertext pairs available,

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

where $q = \lceil \frac{n_1+n_2}{B} \rceil$. The meet-in-the-middle (MTM) attack tries to find the key (K_1, K_2) and works according to Algorithm 1 below.

Algorithm 1 : MTM Attack

1. *Compute $C' = E1_{K_1}(PT_1)$ over all possible values of K_1. Store the pair (C', K_1) and sort according to C'. This step has complexity 2^{n_1} and needs 2^{n_1} memory.*
2. *Compute $C'' = E2^{-1}_{K_2}(CT_1)$ over all possible values of K_2. For all K_2, look for a match for C'' from the pair (C'', K_2) in the stored table. Once any possible key (S_1, S_2) has been identified, test if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for all $i = 2, \ldots, q$. Discard (S_1, S_2) if it does not satisfy this equation for any i. After this step, all the wrong keys will be filtered out, leaving only the correct key with overwhelming probability.*

Note that if $n_2 < n_1$, we could have stored a table of $C' = E2^{-1}_{K_2}(C)$ in step 1 and then computed $C'' = E1_{K_1}(P)$ to search for collision from the table in step 2. This will take up less memory.

The meet-in-the-middle attack on multiple encryption uses 2^{n_1} memory. The attack complexity is as follows:

1. If $n_1 > B$, then

$$\text{attack complexity} = 2^{n_1} + 2^{n_2} + 2^{n_2+(n_1-B)} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1).$$

The first term corresponds to encryption of PT_1 over possible values of K_1. The middle term corresponds to decryption of CT_1 over possible values of

K_2. The magnitude of the last term is due to the fact that for each partially decrypted ciphertext $C'' = E2_{K_2}^{-1}(C_1)$, there will be 2^{n_1-B} keys, S_1, mapping P to C'', so that $2^{n_2+(n_1-B)}$ possible keys will be identified and each has to be verified by $\lceil \frac{n_1+n_2}{B} \rceil - 1$ other plaintext-ciphertext pairs. In this instance, the attack complexity can be subdivided into two cases:
(a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
(b) attack complexity $\approx 2^{n_1}$ if $n_2 \leq B$.

2. If $n_1 \leq B$, then

$$\text{attack complexity} = 2^{n_1} + 2^{n_2} + 2^{n_2} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1)$$

since there will be at most one key, S_1, mapping P to each C''. Therefore,
(a) attack complexity $\approx 2^{n_2}$ if $n_2 > B$; or
(b) attack complexity $\approx \max(2^{n_1}, 2^{n_2})$ if $n_2 \leq B$.

Remark 1. The attack complexity is increased by a factor of at most $\lceil \frac{n_1+n_2}{B} \rceil$ times the approximate complexities we have given in 1(a),(b) and 2(a),(b) above. However, since $\lceil \frac{n_1+n_2}{B} \rceil$ usually lies between 2 and 4, the increase is not significant and hence, we choose not to take it into account in our paper. We will make similar approximations in subsequent derivations.

As can be observed from the discussion above, this attack has complexity much less than the exhaustive search complexity of $2^{n_1+n_2}$ but requires the use of memory.

Example 1. Suppose we apply the MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first DES block with a 56-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining two DES blocks with a 112-bit key. This falls into case 2(a) and hence, the attack complexity is 2^{112}. The memory used is 2^{56}. This is less than the exhaustive search complexity of 2^{168}.

Sometimes when we can afford to have high pre-computation complexity and large memory, we can pre-compute step 1 of the MTM attack by fixing a particular plaintext, P, to speed up the actual attack. In this case, $n_1 > n_2$. The attack complexities are similar to cases 1 and 2 described above, except that this time, we exclude the first term, 2^{n_1}, corresponding to the first step in Algorithm 1 since this is done in the pre-computation. In order to ensure that the attack complexity is always given by 2^{n_2}, we shall assume that the key size of K_1 is always bounded by the effective block size by executing the attack on $\lceil \frac{n_1}{B} \rceil$ blocks of B-bit plaintexts, where B is the true block size of the cipher. Then both the memory required and the pre-computation complexity will be increased to $\lceil \frac{n_1}{B} \rceil \cdot 2^{n_1}$ and the effective block size is $B' = B \cdot \lceil \frac{n_1}{B} \rceil$. In this case, only $\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil$ plaintext-ciphertext pairs are needed for verification of each possible key in step 2 of Algorithm 1. Although we stipulate this requirement only for the MTM attack with pre-computation so far, it shall turn out, as we will see in later sections, that this rule will also apply to all the other attacks in this paper which need pre-computation. Those without pre-computation will not need this requirement. The MTM attack with pre-computation is illustrated in the following example.

Example 2. Suppose we apply the MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. Since $\lceil \frac{n_1}{B} \rceil = 2$, the effective fixed plaintext is taken to be a concatenation of two 64-bit fixed plaintext blocks. We can pre-compute step 1 of the MTM attack with $2 \cdot 2^{112} = 2^{113}$ complexity and store the result in $2 \cdot 2^{112} = 2^{113}$ memory. In this way, we can recover the key from any ciphertext C encrypted from P with complexity 2^{56} by step 2 of the MTM attack.

Note that in the attack of Example 2, we require the ciphertext to be the encryption of a fixed plaintext. This still works in practice because many plaintext messages contain some fixed header, e.g. MIME header, postscript header, pdf header, version number, and we can take this as PT_1.

To avoid confusion, we shall refer to the attack in Example 1 as a MTM attack without pre-computation and the attack in Example 2 as a MTM attack with pre-computation.

4 Applying Guess-and-Determine Method to Meet-in-the-Middle Attack Using Single and Multiple Data (GD-MTM and GDD-MTM respectively)

In [6, Section 7.37], there is a suggestion to independently guess s bits of K_1 for some fixed s $(0 \leq s \leq n_1)$. In this modification of the MTM attack (without pre-computation), the table for K_1 requires $2^{n_1 - s}$ memory for each set of s bits guessed. We call this version from [6] the GD-MTM attack. As before, suppose that there are q plaintext-ciphertext pairs available,

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_q, CT_q)$$

where $q = \lceil \frac{n_1+n_2}{B} \rceil$. The GD-MTM attack is described in Algorithm 2 as follows:

Algorithm 2 : GD-MTM Attack

1. *Fix s bits of K_1 at a particular value.*
2. *Compute $C' = E1_{K_1}(PT_1)$ over all values of K_1 with the s bits fixed as stated in step 1. The table for K_1 will only contain 2^{n_1-s} entries. This step has 2^{n_1-s} time complexity and requires 2^{n_1-s} memory.*
3. *Compute $C'' = E2^{-1}_{K_2}(CT_1)$ over all possible values of K_2. For all K_2, look for a match for C'' from the pair (C'', K_2) in the table. Once any possible key (S_1, S_2) has been identified, test if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for all $i = 2, \ldots, q$. Discard (S_1, S_2) if it does not satisfy this equation for any i.*
4. *Repeat steps 2 and 3 by varying the same s bits over all possible values.*

The memory complexity is 2^{n_1-s} since all computations are done online and the memory is cleared whenever the attacker rebuilds the table for each new guess. The attack complexity is as follows:

1. If $n_1 - s > B$, then

$$\text{attack complexity} = 2^s \left[2^{n_1-s} + 2^{n_2} + 2^{n_2+(n_1-s-B)} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1) \right]$$
$$\approx 2^{n_1} + 2^{n_2+s} + 2^{n_1+n_2-B}$$

   The last term is derived from the fact that for each value of K_1 with s bits fixed and each partially decrypted ciphertext C'', there will be 2^{n_1-s-B} keys, S_1, identified. Again, the attack complexity is given by:
   (a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
   (b) attack complexity $\approx 2^{n_1}$ if $n_2 \leq B$.
2. If $n_1 - s \leq B$, then

$$\text{attack complexity} = 2^s \left[2^{n_1-s} + 2^{n_2} + 2^{n_2} \cdot (\lceil \frac{n_1+n_2}{B} \rceil - 1) \right]$$
$$\approx 2^{n_1} + 2^{n_2+s} + 2^{n_2+s}$$

   Therefore,
   (a) attack complexity $\approx 2^{n_2+s}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1}, 2^{n_2+s})$ if $n_2 \leq B$.

Comparing the usual MTM attack outlined in Section 3 with the GD-MTM attack, it can be observed that if $n_1 \leq B$ and $n_2 > B$, then the GD-MTM attack is effective for building smaller lookup tables using less memory in exchange for more time needed to repeat the procedure.

We propose a simple extension of GD-MTM to GDD-MTM (guess-and-determine MTM attack with multiple data) for the case where a plaintext is encrypted by the block cipher under several keys and the attacker only needs to recover one key out of several keys. Assume that we have $D = 2^d$ encryptions of P using different keys and we are only required to find one of them, $\left(K_1^{(i)}, K_2^{(i)}\right)$ by attacking:

$$C_0 = E_{(K_1^{(0)},K_2^{(0)})}(P), C_1 = E_{(K_1^{(1)},K_2^{(1)})}(P), \ldots, C_{2^d-1} = E_{(K_1^{(2^d-1)},K_2^{(2^d-1)})}(P)$$

where $E_{(K_1,K_2)}(\cdot) = E1_{K_1}(E2_{K_2}(\cdot))$. Also assume that for each $i = 0, 1, \ldots, 2^{d-1}$, we have q' other plaintext-ciphertext pairs available:

$$(PT_1^{(i)}, CT_1^{(i)}), (PT_2^{(i)}, CT_2^{(i)}), \ldots, (PT_{q'}^{(i)}, CT_{q'}^{(i)}),$$

all encrypted using key $(K_1^{(i)}, K_2^{(i)})$, where $q' = \lceil \frac{n_1+n_2}{B} \rceil - 1$. With multiple data applied to the usual MTM attack without pre-computation described in Algorithm 1, C' only needs to be computed over 2^{n_1-d} values of K_1. However, if we also guess s bits of K_1 at the beginning of the attack, then each table of K_1 will contain just 2^{n_1-d-s} entries. The assumption is that one of the s bits guess should correspond to at least one of the 2^d keys with high probability. The process is outlined in the Algorithm 3 below.

Algorithm 3 : GDD-MTM Attack

1. *Fix s bits of K_1 at a particular value.*
2. *Compute $C' = E1_{K_1}(P)$ over 2^{n_1-d-s} values of K_1 with the s bits fixed as stated in step 1. The table for K_1 will only contain 2^{n_1-d-s} entries. This step has time complexity 2^{n_1-d-s} and requires 2^{n_1-d-s} memory.*
3. *Compute $C''_0 = E2^{-1}_{K_2}(C_0)$ over all possible values of K_2. For all K_2, look for a match for C''_0 from the pair (C''_0, K_2) in the table. Once any possible key (S_1, S_2) has been identified, test if $E2_{S_2}(E1_{S_1}(PT_i)) = CT_i$ for $\lceil \frac{n_1+n_2}{B} \rceil - 1$ other plaintext-ciphertext pairs (PT_i, CT_i). Discard (S_1, S_2) if it does not satisfy this equation for any i.*
4. *If the correct key $(K_1^{(0)}, K_2^{(0)})$ is not found, then repeat steps 3 and 4 for $j = 1, \ldots, 2^d - 1$ consecutively until one correct key is found. Note that the set of $\lceil \frac{n_1+n_2}{B} \rceil - 1$ plaintext-ciphertext pairs used for verification need not be uniform across all the different C_j.*
5. *If no correct key is found by step 4, vary the same s bits of K_1 over the rest of the 2^s possibilities and repeat steps 1 to 4 until a correct key is found.*

In this case, the memory complexity is 2^{n_1-d-s}. The attack complexity is given as follows:

1. If $n_1 - d - s > B$, then

$$\begin{aligned} &\text{attack complexity} \\ &= 2^s \left[2^{n_1-d-s} + 2^d \left(2^{n_2} + 2^{n_2+(n_1-d-s-B)} \cdot (\lceil \tfrac{n_1+n_2}{B} \rceil - 1)\right)\right] \\ &\approx 2^{n_1-d} + 2^{n_2+d+s} + 2^{n_1+n_2-B} \end{aligned}$$

   The two subcases are:
   (a) attack complexity $\approx 2^{n_1+n_2-B}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1-d}, 2^{n_1+n_2-B})$ if $n_2 \leq B$.
2. If $n_1 - d - s \leq B$, then

$$\begin{aligned} &\text{attack complexity} \\ &= 2^s \left[2^{n_1-d-s} + 2^d \left(2^{n_2} + 2^{n_2} \cdot (\lceil \tfrac{n_1+n_2}{B} \rceil - 1)\right)\right] \\ &\approx 2^{n_1-d} + 2^{n_2+d+s} + 2^{n_2+d+s} \end{aligned}$$

   Therefore,
   (a) attack complexity $\approx 2^{n_2+d+s}$ if $n_2 > B$; or
   (b) attack complexity $\approx \max(2^{n_1-d}, 2^{n_2+d+s})$ if $n_2 \leq B$.

Example 3. Now let us apply the GD-MTM attack to the 168-bit triple DES encryption of a fixed plaintext P where $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first DES block with a 56-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining two DES blocks with a 112-bit key. Assume that our system has a memory limitation of 2^{48}, that is, $s = 8$. Then the time complexity is $2^{112+8} = 2^{120}$.

Example 4. In this example, we shall apply the GDD-MTM attack to 168-bit triple DES encryption of a fixed plaintext P to recover one key out of a key-pool of 2^{14} keys, that is, $d = 14$. We take $E1_{K_1}(\cdot)$ to be encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ to be encryption over the remaining DES block with a 56-bit key. Suppose we can afford 2^{64} memory so that s is fixed at 34. Then the time complexity is given by $\max(2^{112-14}, 2^{56+14+34}) = 2^{104}$.

Comparing Examples 3 and 4, we see that with multiple data, the attacker can achieve lower time complexity at the expense of more memory requirement.

5 Applying Time-Memory Trade-Off to Meet-in-the-Middle Attack (TMTO-MTM and Rainbow-MTM)

In this section, we apply the time-memory trade-off (TMTO) attack of [5] to the MTM attack with pre-computation and we call it the TMTO-MTM attack. Basically, we apply the TMTO attack to the pre-computation step 1 of the MTM attack. Again we have the same restriction as Example 2, i.e. we require the ciphertext to be the encryption of a fixed plaintext P. Furthermore, for this attack and subsequent ones (Rainbow-MTM, BS-MTM, and TMD-MTM), all of which require pre-computation, we shall also assume that the key size of K_1 is always bounded by the effective block size, B'. This is done by attacking $\lceil \frac{n_1}{B} \rceil$ blocks of B-bit plaintexts, where B is the effective block size of the cipher. Then $B' = B \cdot \lceil \frac{n_1}{B} \rceil$ and both the memory and pre-computation complexities are increased by a factor of $\lceil \frac{n_1}{B} \rceil$. Let E' be defined by

$$E'_{(K_1,K_2)}(P_1 \parallel \ldots \parallel P_{\lceil \frac{n_1}{B} \rceil}) = E2_{K_2}(E1_{K_1}(P_1)) \parallel \ldots \parallel E2_{K_2}(E1_{K_1}(P_{\lceil \frac{n_1}{B} \rceil})).$$

$E1'$ and $E2'$ are also defined in a similar way.

For the TMTO-MTM attack, assume that we have $C = E'_{(K_1,K_2)}(P)$ (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ fixed plaintext blocks). Also assume that we have q'' other plaintext-ciphertext pairs available:

$$(PT_1, CT_1), (PT_2, CT_2), \ldots, (PT_{q''}, CT_{q''}),$$

all encrypted with E using key (K_1, K_2), where $q'' = \lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil$. The algorithm is shown in Algorithm 4.

Algorithm 4 : TMTO-MTM Attack

1. *Pre-processing:*
   (a) *Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ plaintext blocks) and let $C = E2'_{K2}(E1'_{K1}(P))$. Define a one-way function $g(K) = E1'_K(P)$.*

(b) *Form t tables of size $m \times t$ as follows: For each table, randomly choose m distinct start points $X_{i,0}$, $i = 0, 1, \ldots, m-1$ and compute m chains of values of length t, $X_{i,1} = f(X_{i,0})$, $X_{i,2} = f(X_{i,1})$, $\ldots$, $X_{i,t} = f(X_{i,t-1})$, where $f(x) = h \circ g(x)$ and h is a simple reversible modification of the output of f (e.g. bit shuffling) if $n_1 = B'$; otherwise, h is a truncation from B' bits to n_1 bits followed by a simple transformation if $n_1 < B'$. This will form a table of size $m \times t$. Repeat this process to form t such tables where all the start points are distinct. Each h should be different for all t tables. We expect to cover most of the key space of K_1 by this process, which has complexity $mt^2 \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil$.*

(c) *To reduce memory requirements, discard all intermediate points and sort the start and end points $(X_{i,0}, X_{i,t})$ according to the end points $X_{i,t}$. Store the start and end points of each table using $mt \cdot \lceil \frac{n_1}{B} \rceil$ memory.*

2. *Attack:*

(a) *Compute $C' = h(E2'^{-1}_{K_2}(C))$ over all possible values of K_2.*

(b) *For a particular K_2, check to see if C' is equal to an end-point $X_{i,t}$ of a table. If it is, then we can guess that $(X_{i,t-1}, K_2)$ is a possible encryption key. The value $X_{i,t-1}$ can be computed from $f^{t-1}(X_{i,0})$. Check whether $E2'^{-1}_{K_2}(C) = E1'_{X_{i,t-1}}(P)$ to see if $(X_{i,t-1}, K_2)$ is a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1}, K_2)$ if it does not satisfy this equation for any i.*

(c) *If not, compute $f^j(C')$, $j = 1, \ldots, t-1$ and check to see if it is equal to an end-point $X_{i,t}$ of a table. If it is, then $(X_{i,t-1-j}, K_2)$ is a possible encryption key. The value $X_{i,t-1-j}$ can be computed from $f^{t-1-j}(X_{i,0})$. Again, check whether $E2'^{-1}_{K_2}(C) = E1'_{X_{i,t-1-j}}(P)$ to see if $(X_{i,t-1-j}, K_2)$ is indeed a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1-j}}(PT_i)) = CT_i$ for $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1-j}, K_2)$ if it does not satisfy this equation for any i. The complexity of covering a table (excluding verification) is $t \cdot \lceil \frac{n_1}{B} \rceil$.*

(d) *If the key (K_1, K_2) is not found in a table, then repeat steps 2(b) and 2(c) for the other $t-1$ tables to find the key. Thus the total complexity of covering these tables for all keys K_2 (excluding verification) is $t^2 \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil$.*

As we have noted, the pre-processing complexity in step 1 is $2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil = mt^2 \cdot \lceil \frac{n_1}{B} \rceil$. Assuming we use a memory of $2^{mem} \cdot \lceil \frac{n_1}{B} \rceil = mt \cdot \lceil \frac{n_1}{B} \rceil$ for our attack, the attack complexity in step 2 is

$$\begin{aligned}
&t^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (mt^2/mt)^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (2^{n_1}/2^{mem})^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= 2^{2(n_1-mem)+n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&\approx 2^{2(n_1-mem)+n_2} + 2^{n_2}.
\end{aligned}$$

Example 5. Suppose as in Example 2, we apply the TMTO-MTM attack on triple DES where $E1_{K_1}(\cdot)$ is encryption over the first two DES blocks with 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with 56-bit keys. The effective fixed plaintext is taken to be a concatenation of two 64-bit fixed plaintext blocks. Suppose we can only afford 2^{101} instead of 2^{113} memory. Then the pre-processing complexity of TMTO-MTM is 2^{113} but the attack complexity is now $2^{2(112-100)+56} = 2^{80}$ instead of 2^{56}.

Thus the TMTO-MTM attack is a trade-off of less memory at the expense of more attack complexity for the pre-computed MTM attack.

In [8], Oechslin published a new way of forming the pre-computation table using rainbow chains. In this method, only one table is required with mt starting points and chains of length t, where $mt^2 = \textit{size of key space}$. Each chain uses t reduction functions, f_i, starting with reduction function f_1 and ending with reduction function f_t. If two chains collide, they merge only if the collision appears at the same position in both chains. The probability of success of the rainbow attack was found to be approximately equal to the success probability of t classical tables of size $m \times t$. This method reduces the number of table look-ups by a factor of t compared to the Hellman's original TMTO method. Rainbow chains eliminate the occurrence of loops and any merges amongst chains are detectable. Furthermore, they reduce the total complexity of the attack by a factor of 2.

We may apply the rainbow attack to the precomputation step of the MTM attack in a way analogous to how the TMTO-MTM attack was constructed. We call this the Rainbow-MTM attack. More specifically, the process is given in Algorithm 5.

Algorithm 5 : Rainbow-MTM Attack

1. *Pre-processing:*
   (a) *Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.*
   (b) *Form a table of size $mt \times t$ as follows: Randomly choose mt distinct start points $X_{i,0}$, $i = 0, 1, \ldots, mt-1$ and compute mt chains of values of length t, $X_{i,1} = f_1(X_{i,0})$, $X_{i,2} = f_2(X_{i,1})$, $\ldots$, $X_{i,t} = f_t(X_{i,t-1})$, where $f_i(x) = h_i \circ g(x)$ and h_i is a simple reversible modification of the output of f (e.g. bit shuffling) if $n_1 = B'$; otherwise, h_i is a truncation from B' bits to n_1 bits followed by a simple transformation if $n_1 < B'$. This will form a table of size $mt \times t$. Ensure that all the mt endpoints are distinct. This pre-processing step has complexity $mt^2 \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1} \cdot \lceil \frac{n_1}{B} \rceil$.*
   (c) *Sort the start and end points $(X_{i,0}, X_{i,t})$ according to the end points $X_{i,t}$. Store them using $mt \cdot \lceil \frac{n_1}{B} \rceil$ memory.*
2. *Attack:*
   (a) *Compute $C' = E2'^{-1}_{K_2}(C)$ over all possible values of K_2.*
   (b) *For a particular K_2, check to see if $h_t(C')$ is equal to an end-point $X_{i,t}$ of a table. If it is, then we can guess that $(X_{i,t-1}, K_2)$ is a possible encryption key. The value $X_{i,t-1}$ can be computed by rebuilding the chain from the corresponding start point $X_{i,0}$. Check whether $C' = E1'_{X_{i,t-1}}(P)$*

to see if $(X_{i,t-1}, K_2)$ is indeed a possible encryption key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1}}(PT_i)) = CT_i$ for $\left(\lceil\frac{n_1+n_2}{B}\rceil - \lceil\frac{n_1}{B}\rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1}, K_2)$ if it does not satisfy this equation for any i.

(c) *If the correct key is not found, compute $f_t \circ f_{t-1} \circ \ldots \circ f_{t-j+1} \circ h_{t-j}(C')$, $j = 1, \ldots, t-1$ and check to see if it is equal to an end-point $X_{i,t}$. If it is, then $(X_{i,t-1-j}, K_2)$ is a possible encryption key. The value $X_{i,t-1-j}$ can be computed from the start point $X_{i,0}$. Again, check whether $C' = E1'_{X_{i,t-1-j}}(P)$ to see if $(X_{i,t-1-j}, K_2)$ is indeed a possible key. If it is, test if $E2_{K_2}(E1_{X_{i,t-1-j}}(PT_i)) = CT_i$ for $\left(\lceil\frac{n_1+n_2}{B}\rceil - \lceil\frac{n_1}{B}\rceil\right)$ other plaintext-ciphertext pairs. Discard $(X_{i,t-1-j}, K_2)$ if it does not satisfy this equation for any i.The total complexity of covering this table for all keys K_2 (excluding verification) is $\frac{t(t+1)}{2} \cdot 2^{n_2} \cdot \lceil\frac{n_1}{B}\rceil \approx \frac{t^2}{2} \cdot 2^{n_2} \cdot \lceil\frac{n_1}{B}\rceil$.*

So for this attack, as compared to the TMTO-MTM attack, the time complexity is reduced slightly to $2^{2(n_1-mem)+n_2-1} \cdot \lceil\frac{n_1}{B}\rceil + 2^{n_2}$ while the memory and pre-computation requirements remain the same.

Remark 2. In [1, Section 6.1], there was a criticism levelled against rainbow chains. According to them, in the Hellman scheme, it is possible to store just half the number of bits of the start and end points compared to the Rainbow scheme, reducing time complexity by a factor of 4 and therefore offsetting the claimed improvement of the Rainbow scheme which only reduces the time complexity by a smaller factor of 2. However, we believe that rainbow chains still offer other important advantages as given in [8, Section 3].

Remark 3. In [7], the authors suggested another method of dealing with the case when the key size is greater than the block size of the cipher. For their attack, the pre-processing stage computes 2^{n_1-B} tables, where each row of a particular table starts with a B-bit string which is then concatenated with an $(n_1 - B)$-bit fixed string to encrypt the zero vector. The result is then concatenated with the fixed string again and the same procedure is repeated to obtain a chain of values. Note, however, that they assume the more stringent requirement that $B < n_1 \leq 2B$.

6 Applying Biryukov-Shamir Time-Memory-Data Trade-Off to Meet-in-the-Middle Attack (BS-MTM)

Let us consider a scenario where a plaintext is encrypted by a block cipher under several keys. One such example was suggested by Biryukov et al. in [3] where they attacked the Unix password encryption scheme. Because they only need to recover one key out of several keys, they can apply the time-memory-data (TMD) trade-off attack of [2]. To distinguish between this attack and a more general attack that we will discuss in the next section, we will refer to this attack as the BS attack. In the BS attack, the complexity of pre-processing can be reduced from N to N/D where N is the key space and D is the size of the keypool from which one needs to be found.

In this section, we shall try to recover the key from ciphertexts encrypted from a fixed plaintext P by applying the BS attack to the MTM attack with pre-computation. Our new attack is called the BS-MTM attack. Assume, as in Section 4 that we have $D = 2^d$ encryptions of P (where P is a concatenation of $\lceil \frac{n_1}{B} \rceil$ fixed plaintext blocks) using different keys and we only need to find one of them, i.e. find one of $(K_1^{(i)}, K_2^{(i)})$ by attacking:

$$C_0 = E'_{(K_1^{(0)},K_2^{(0)})}(P), C_1 = E'_{(K_1^{(1)},K_2^{(1)})}(P), \ldots, C_{2^d-1} = E'_{(K_1^{(2^d-1)},K_2^{(2^d-1)})}(P),$$

where $E'_{(K_1,K_2)}(\cdot) = E1'_{K_1}(E2'_{K_2}(\cdot))$. This means that the effective block size to be $B' = B \cdot \lceil \frac{n_1}{B} \rceil \geq n_1$. Also assume that for each $i = 0, 1, \ldots, 2^{d-1}$, we have q'' other plaintext-ciphertext pairs available:

$$(PT_1^{(i)}, CT_1^{(i)}), (PT_2^{(i)}, CT_2^{(i)}), \ldots, (PT_{q''}^{(i)}, CT_{q''}^{(i)}),$$

all encrypted with E using key $(K_1^{(i)}, K_2^{(i)})$, where $q'' = \lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil$. Algorithm 6 illustrates the attack.

Algorithm 6 : BS-MTM Attack

1. *Pre-processing:*
   (a) *Choose two positive integers m, t such that $2^{n_1} = mt^2$. Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.*
   (b) *Form $t/2^d$ tables of size $m \times t$ as in step 1(b) of Algorithm 4 with complexity $mt^2/2^d \cdot \lceil \frac{n_1}{B} \rceil = 2^{n_1-d} \cdot \lceil \frac{n_1}{B} \rceil$. This will cover $1/2^d$ of the keyspace of K_1.*
   (c) *Sort and store the start and end points of each table using $mt/2^d \cdot \lceil \frac{n_1}{B} \rceil$ memory as in step 1(c) of Algorithm 4.*
2. *Attack:*
   (a) *Compute $C'_0 = h(E2'^{-1}_{K_2}(C_0))$ over all possible values of K_2.*
   (b) *Search to see if $C'_0, f(C'_0), \ldots, f^{t-1}(C'_0)$ is equal to one of the end points of any of our stored tables and compute the key K_1 from the corresponding start point as in steps 2(b), 2(c) and 2(d) of Algorithm 4. The complexity of covering these tables for all keys K_2 (excluding verification) is $t/2^d \cdot t \cdot 2^{n_2} \cdot \lceil \frac{n_1}{B} \rceil$.*
   (c) *If the correct key $\left(K_1^{(0)}, K_2^{(0)}\right)$ is not contained in the space of 2^{n_1-d} keys computed, then proceed to repeat the attack for C_j, $j = 1, \ldots, 2^d - 1$ consecutively until one correct key is found. Note that the set of $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ plaintext-ciphertext pairs used for verification phase need not be uniform across all the different C_j.*

As we have noted, the pre-processing complexity in step 1 is $2^{n_1-d} \cdot \lceil \frac{n_1}{B} \rceil$. Assuming we use a memory of $2^{mem} \cdot \lceil \frac{n_1}{B} \rceil = mt/2^d \cdot \lceil \frac{n_1}{B} \rceil$ for our attack, the attack complexity in step 2 is

$$\begin{aligned}
&2^d \cdot t^2/2^d \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^d \cdot 2^{n_2} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (mt^2/mt)^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= (2^{n_1}/(2^{mem} \times 2^d))^2 \cdot 2^{n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&= 2^{2(n_1-d-mem)+n_2} \cdot \lceil \tfrac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \tfrac{n_1+n_2}{B} \rceil - \lceil \tfrac{n_1}{B} \rceil\right) \\
&\approx 2^{2(n_1-d-mem)+n_2} + 2^{n_2+d}.
\end{aligned}$$

Example 6. Suppose we want to attack the 168-bit triple DES encryption of a fixed plaintext P and we only need to recover one key out of a key pool of 2^{14} keys. We apply the BS-MTM attack where $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. If we use 2^{85} memory, then the pre-computation complexity is $2^{112-14} \cdot 2 = 2^{99}$ and the attack complexity is $2^{2(112-14-84)+56} = 2^{84}$. In comparison, according to [3, Table 2], a direct application of the time-memory-data trade-off attack on triple DES where we recover one key out of a pool of 2^{42} keys requires: 2^{84} memory, 2^{126} pre-computation and 2^{84} attack complexity. Therefore, our BS-MTM attack achieves lower data and pre-processing complexity than the BS attack.

7 Applying TMTO - Data Curve to Meet-in-the-Middle Attack (TMD-MTM)

In [3], the authors also presented a unifying framework for the analysis of multiple data trade-offs. The BS attack adapted from [2] is considered as a special case of this more general framework. Furthermore, they identified a new class of single table multiple data trade-offs which cannot be obtained from the BS attack. In this section, we shall apply their more general TMD attack to the MTM attack with pre-computation and we call our new attack the TMD-MTM attack. The mode of this attack follows a similar procedure to the BS-MTM attack as outlined in Algorithm 6. The main difference lies in the use of the birthday bound as will be highlighted later. Suppose, as before, that $B' = B \cdot \lceil \frac{n_1}{B} \rceil \geq n_1$. The attack is given by Algorithm 7 as follows:

Algorithm 7 : TMD-MTM Attack

1. *Pre-processing:*
   (a) *Fix a plaintext P and define a one-way function $g(K) = E1'_K(P)$.*
   (b) *Form r tables of size $m \times t$ as in step 1(b) of algorithm 4. The parameters (r, m, t) are chosen such that the tables will cover $1/2^d$ of the keyspace of K_1 (where $D = 2^d$ is the number of encryptions of P using different keys). The exact conditions that they must satisfy will be given below.*
   (c) *Sort and store the start and end points of each table using $rm \cdot \lceil \frac{n_1}{B} \rceil$ memory as in step 1(c) of Algorithm 4.*

2. *Attack:*
   (a) *Compute* $C_0' = h(E2'^{-1}_{K_2}(C_0))$ *over all possible values of* K_2.
   (b) *Search to see if* $C_0', f(C_j'), \ldots, f^{t-1}(C_0')$ *is equal to one of the end points of any of our stored tables and compute the key* K_1 *from the corresponding start point as in steps 2(b), 2(c) and 2(d) of Algorithm 4.*
   (c) *If the correct key* $\left(K_1^{(0)}, K_2^{(0)}\right)$ *is not contained in the space of* 2^{n_1-d} *keys computed, then proceed to repeat the attack and verification for* C_j, $j = 1, \ldots, 2^d - 1$ *consecutively until one correct key is found. Note that the set of* $\left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right)$ *plaintext-ciphertext pairs used for verification phase need not be uniform across all the different* C_j.

In our TMD-MTM attack, we shall assume that the number of columns of each table is $\gg 1$ and

$$\text{time for one table look-up} \approx \text{time for one invocation of } f.$$

Then we have the relations:

$$\left.\begin{array}{rl}
N & = 2^{n_1} \\
PC & = rmt \quad (\#\ g \text{ invocations in the pre-computation phase}) \\
 & = \frac{N}{D} \quad (\text{coverage}) \\
PC' & = PC \cdot \lceil \frac{n_1}{B} \rceil \quad (\text{pre-computation complexity}) \\
M & = rm \\
M' & = M \cdot \lceil \frac{n_1}{B} \rceil \quad (\text{memory}) \\
D & = 2^d \quad (\#\ \text{encryptions of } P \text{ using different keys}) \\
T & = rtD \\
T' & = 2^{n_2} \cdot T \cdot \lceil \frac{n_1}{B} \rceil + 2^{n_2+d} \cdot \left(\lceil \frac{n_1+n_2}{B} \rceil - \lceil \frac{n_1}{B} \rceil\right) \\
 & \approx 2^{n_2+d} rt + 2^{n_2+d} \quad (\text{time for online phase}) \\
mt^2 & \leq N \quad (\text{birthday bound}) \\
T' & < PC'
\end{array}\right\} \qquad (1)$$

The last inequality in (1) was added since in practical attacks, we usually require the online attacking time to be less than the offline table preparation time. We can solve for r, m and t to get:

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

Based on these, we can derive the TMTO curve given by:

$$\left.\begin{aligned} D &= N^w \\ MT &= N^x \\ M &= N^y \\ mt^2 &= N^z \\ PC &= N^{1-w} \\ T &= N^{x-y} \end{aligned}\right\} \qquad (3)$$

where

$$\left.\begin{gathered} 2w + x + y + z = 3 \\ 0 \le w < 1 \\ 0 \le y, x - y < 1 \le x \\ w + y \le 1 \\ 0 \le z \le 1 \\ \tfrac{n_2}{n_1} < 1 - (w + x) + y \end{gathered}\right\} \qquad (4)$$

Therefore, any set of parameters (w, x, y, z) satisfying (4) gives a valid attack. We can also express (r, m, t) in terms of (w, x, y, z) as follows:

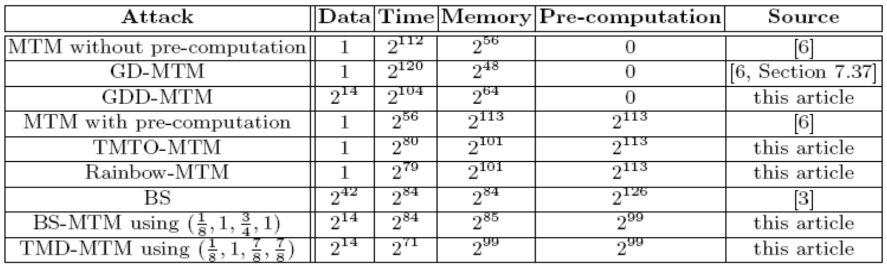
$$\left.\begin{aligned} r &= N^{x-1} \\ m &= N^{1-(x-y)} \\ t &= N^{1-w-y} \end{aligned}\right\} \qquad (5)$$

It is now easy to see that the BS-MTM attack is a special case of the TMD-MTM attack with $z = 1$, i.e. $mt^2 = N$. In Example 6, we used the parameters $(w, x, y, z) = (\frac{1}{8}, 1, \frac{3}{4}, 1)$.

Example 7. Now let us apply the TMD-MTM attack to the 168-bit triple DES encryption of a fixed plaintext P to recover one key out of a keypool of 2^{14} keys. In this case $w = \frac{1}{8}$. As before, $E1_{K_1}(\cdot)$ is encryption of a fixed plaintext P over the first two DES blocks with a 112-bit key and $E2_{K_2}(\cdot)$ is encryption over the remaining DES block with a 56-bit key. In order to achieve minimum attack complexity $T' = 2^{56} \cdot N^{x-y} + 2^{70}$, we take x to be minimum (i.e. $x = 1$) and y to be maximum (i.e. $y = 1 - w = \frac{7}{8}$). Then $z = 3 - (2w + x + y) = \frac{7}{8}$. This gives $M' = 2^{99}$, $PC' = 2^{99}$ and $T' = 2^{71}$ with parameters $(w, x, y, z) = (\frac{1}{8}, 1, \frac{7}{8}, \frac{7}{8})$. Since $r = x = 1$, only 1 table is needed. Comparing with the BS-MTM attack in Example 6, this TMD-MTM attack has lower attack complexity at the expense of more memory required.

8 Conclusion

In this paper, we presented one new no pre-computation attack using the guess-and-determine technique for multiple data — the GDD-MTM attack — improvising the previously known GD-MTM attack for single data. We have also proposed four new attacks involving pre-computation — the TMTO-MTM, Rainbow-MTM, BS-MTM, and TMD-MTM attacks — by applying the TMTO, Rainbow, BS, and TMD attacks respectively to the MTM attack. Figure 1 below gives a comparison of the time-memory-data trade-offs of the attacks on triple DES.



Attack	Data	Time	Memory	Pre-computation	Source
MTM without pre-computation	1	2^{112}	2^{56}	0	[6]
GD-MTM	1	2^{120}	2^{48}	0	[6, Section 7.37]
GDD-MTM	2^{14}	2^{104}	2^{64}	0	this article
MTM with pre-computation	1	2^{56}	2^{113}	2^{113}	[6]
TMTO-MTM	1	2^{80}	2^{101}	2^{113}	this article
Rainbow-MTM	1	2^{79}	2^{101}	2^{113}	this article
BS	2^{42}	2^{84}	2^{84}	2^{126}	[3]
BS-MTM using $(\frac{1}{8}, 1, \frac{3}{4}, 1)$	2^{14}	2^{84}	2^{85}	2^{99}	this article
TMD-MTM using $(\frac{1}{8}, 1, \frac{7}{8}, \frac{7}{8})$	2^{14}	2^{71}	2^{99}	2^{99}	this article

Fig. 1. Attacks on 3-key triple DES

Remark 4. For all the attacks with pre-computation in Figure 1 except the BS attack, the attacks are done on two blocks of fixed plaintext.

As can be observed, the attacks without pre-computation generally have higher attack complexities but require less memory. The attacks with pre-computation can afford much lower online attack complexities but require larger memory and pre-computations. Another difference is that the no pre-computation attacks are known plaintext attacks while the pre-computation attacks are chosen plaintext attacks. In both types of attacks, the presence of multiple data can help reduce the time/memory/pre-computation complexities.

Our proposed attacks provide viable methods to achieve new time-memory-data trade-offs apart from previously known attacks. In particular, our new attacks involving pre-computation are desirable as they can achieve lower data and pre-computation complexity than the attacks suggested by Biryukov [3].

References

1. E. Barkan, E. Biham, and A. Shamir, "Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs", LNCS 4117, *CRYPTO 2006*, pp. 1-21, Springer, 2006.
2. A. Biryukov and A. Shamir, "Cryptanalytic Time/Memory/Data Trade-offs for Stream Ciphers", LNCS 1976, *Asiacrypt 2000*, pp. 1-13, Springer-Verlag, 2000.
3. A. Biryukov, S. Mukkhopadhyay, and P. Sarkar, "Improved Time-Memory Trade-Off with Multiple Data", LNCS 3897, *Selected Areas in Cryptography 2005*, pp. 110-127, Springer-Verlag, 2005.
4. W. Diffie and M. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", Computer 10 (6), pp. 74-84, June 1977.
5. M. Hellman, "A Cryptanalytic Time-Memory Trade-Off", *IEEE Trans. on Information Theory*, vol. 26, pp. 401-406, 1980.

6. A. Menezes, P. C. van Oorshot, and S. Vanstone, Chapter 7, *Handbook of Applied Cryptography*, CRC Press, 1996.
7. M. Mihaljevic, M. Fossorier, and H. Imai, “Security Evaluation of Certain Broadcast Encryption Schemes Employing a Generalized Time-Memory-Data Trade-off”, *IEEE Communication Letters*, vol. 11, pp. 988-990, Dec. 2007.
8. P. Oechslin, “Making a Faster Cryptanalytic Time-Memory Trade-off”, LNCS 2729, *Advances in Cryptology - CRYPTO 2003* (D. Boneh, ed.), pp. 617-630, Springer-Verlag, 2003.