Attacking Triple Encryption

Stefan Lucks*

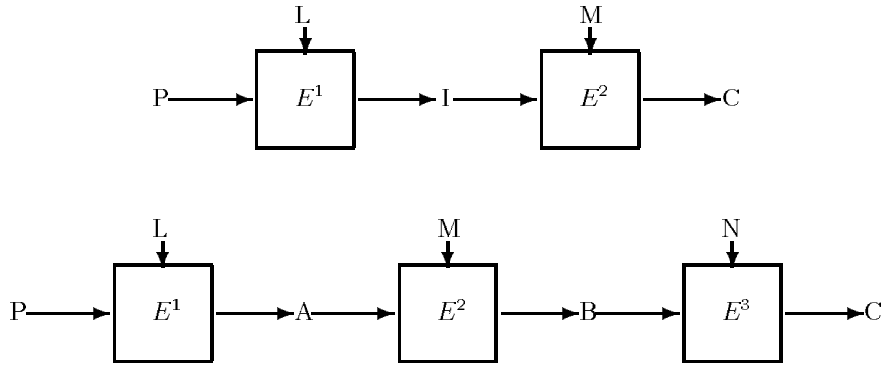
Theoretische Informatik
Universität Mannheim
68131 Mannheim A5, Germany
(email: lucks@pi3.informatik.uni-mannheim.de)

Abstract. The standard technique to attack triple encryption is the meet-in-the-middle attack. In this paper, more efficient attacks are presented. Compared to meet-in-the-middle, our attacks either greatly reduce the number of single encryptions to be done, or somewhat reduce the overall number of steps. Especially, about 2^{108} steps of computation are sufficient to break three-key triple DES. If one concentrates on the number of single DES operations and assumes the other operations to be much faster, 2^{90} of these are enough. We use this to compare the security of triple DES and DESX.

1 Introduction

The most well-known symmetric encryption algorithm is the Data Encryption Standard (DES). It defines a block cipher with 64-bit blocks and 56-bit keys. Due to questions raised regarding the small key size, several varieties of multiple encryption have been considered for the DES, including double and triple DES.

Fig. 1. Double encryption (top) and triple encryption (bottom)



* A part of this research was done while the author was at the University of Göttingen.

In this paper, we consider arbitrary single encryption functions $E\{0,1\}^k \times \{0,1\}^s \longrightarrow \{0,1\}^s$ with k-bit keys and a block size of s bits; but we also point out the consequences of our findings for triple DES. Since multiple encryption is mainly of relevance to strengthen block ciphers with a small key space, we concentrate on $k \leq s$. With two k-bit keys L and M and two encryption functions E^1 and E^2, double encryption is defined by $C = E^2_M(E^1_L(P))$. Here, C denotes the ciphertext and P the plaintext. Similarly, triple encryption is defined by $C = E^3_N(E^2_M(E^1_L(P)))$. Figure 1 describes double and triple encryption. If $L = N$, this defines the special case of two-key triple encryption. In this paper, we concentrate on the case of general (three-key) triple encryption.

Double DES is double encryption with $E^1 = E^2 = E$, triple DES is usually defined by $E^1 = E^3 = E$, $E^2 = D$, where E denotes the (single) DES encryption function and D its decryption counterpart.

In general, we assume the functions E^i and D^i to behave like a set of 2^k random permutations E^i_K with $K \in \{0,1\}^k$, chosen according to the uniform probability distribution. In general, nonrandom statistical properties are considered to be weaknesses of block ciphers. In the special case of the DES, two important statistical weaknesses are known, the complementation property, which is exploited in Section 6 of this paper, and a small number of weak keys.

All attacks considered in this paper are key-search attacks and exploit known (or chosen) pairs of plaintext and ciphertext. To measure the complexity of an attack, we consider four values:

1. The number of known plaintext-ciphertext pairs.
2. The storage space required for the attack.
3. The number of single encryptions $y := E^i_K(x)$ or $x := D^i_K(y)$ to mount the attack.
4. The overall number of operations ("steps") to mount the attack.

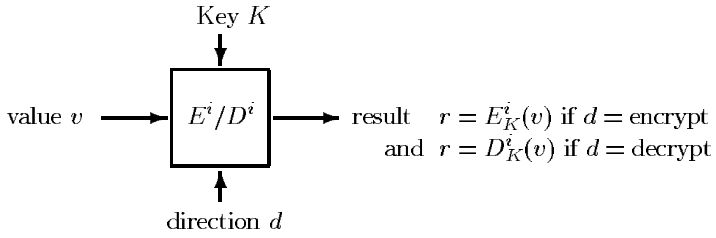
The third value demands some explanation: Clearly, given a key K and a plaintext x (or a ciphertext y) the attacker can compute the corresponding ciphertext $y := E^i_K(x)$ (or the corresponding plaintext). A good block cipher behaves like a random permutation, hence given some triples (plaintext,ciphertext,key) one can't find other triples more efficiently than by encrypting/decrypting again.

Attacking multiple encryption without breaking the underlying encryption function can be described as attacking multiple encryption in the presence of encryption/decryption oracles. Figure 2 visualizes such an oracle. The underlying cipher is treated as a black box. We simply write "single encryption" for accessing the encryption/decryption oracle. Much work has been done with respect to this model.

This view also motivates to specifically count the single encryptions, in addition to counting all steps. Note that such a single encryption counts as one step, but in practice is an exceptionally complex step by itself, compared to common operations like comparisons and table look-ups.

One may as well concentrate on the number of single encryptions and disregard the number of steps and the amount of space required. This is an approved

Fig. 2. An encryption/decryption oracle



method for estimating the minimum strength of a composed cipher. In this context, one investigates the soundness of the composition technique itself and neglects possible weaknesses of the underlying encryption functions. Idealizations of the underlying encryption functions are accessible to the attacker by querying encryption/decryption oracles, but the attacker has no knowledge about the oracles' internals. In contrast to real ciphers, the attacker can't replace an oracle query by a couple of simple steps. In the sequel, we refer to this point of view as the "black-box-only" model.

The rest of this paper is organized as follows. Section 2 describes previously known attacks, concentrating on the meet-in-the-middle attack. In Section 3, we introduce the notion of "t-collisions" and use it for a technique to reduce the number of steps. In Sections 4 and 5 we consider single encryptions to be much slower than each of the other steps, and we design attacks optimized to save single encryptions (but not the total number of steps); Section 5 also regards the black-box-only model. In Section 6 we exploit the complementation property of DES and triple DES to further improve our attacks. Finally, in Section 7 we concentrate on the consequences of our findings for the security of triple DES.

2 Previous Work

Double encryption can be broken with meet-in-the-middle (MITM). This attack requires $\lceil 2k/s \rceil$ known plaintext/ciphertext pairs on the average, about 2^k units of storage, about 2^k single encryptions, and about as much steps. For a plaintext p and a corresponding ciphertext c, compute all values $I_L = E^1_L(p)$ and store all pairs (I_L, L) in a table, indexed by I_L. Since there are 2^k keys L, this requires 2^k units of storage, 2^k steps and 2^k single encryptions. Now, all values $I'_M = D^2_M(c)$ are computed. For the correct key pair (L, M) the equation $I_L = I'_M$ must hold. Thus the attacker needs to look up I'_M in the previously computed table of pairs (I_L, L).

Two-key triple encryption can be broken by a chosen plaintext attack using about 2^k units of everything: 2^k plaintext/ciphertext pairs, 2^k units of storage, 2^k single encryptions, and 2^k steps, see [6].

The best known way to attack general triple encryption is also by MITM [5, Section 7.2.3]. Let a plaintext/ciphertext pair (p, c) be given. Proceed as follows:

1. Compute all values $b_N = D_N^3(c)$, $N \in \{0,1\}^k$, and store the pairs (b_N, N) in a table, indexed by b_N.
2. Compute all values $b_{L,M} = E_M^2(E_L^1(p))$ with $L, M \in \{0,1\}^k$, and look for $(b_{L,M}, N)$ in the previously computed table of pairs (b_N, N).[1]
3. Test all key triples (L, M, N) with $b_{L,M} = b_N$ until only one such triple remains.

The first stage requires about 2^k steps and single encryptions and as much units of storage. The second requires about 2^{2k} steps and single encryptions. The third stage is cheap. Note that we need at least $l \geq \lceil 3k/s \rceil$ pairs of plaintext and ciphertext for the attack. In the case of triple DES, we need $l \geq 3 = \lceil 3 * 56/64 \rceil$ such pairs, about 2^{56} units of storage, about 2^{112} single encryptions and the same number of steps (mainly table look-ups). (The *expected* number of steps and single encryptions needed for the MITM attack is 2^{111}. This is the number we use when comparing the MITM attack with our probabilistic attacks.)

Advanced MITM techniques for attacking two-key triple encryption have been studied by van Oorschot and Wiener [7]. The same authors also proposed advanced MITM techniques for attacking double encryption [8].

Kelsey, Schneier, and Wagner [2] demonstrated how to attack three-key triple DES using related-key techniques. Let a plaintext p and a corresponding ciphertext c be known to the attacker. Assume the attacker to be able to change the first subkey from L to $L + \Delta$ (both L and $L + \Delta$ unknown to the her, but Δ known). If the attacker receives the decryption of c under the modified key, then she can find the subkey L using only 2^k steps (and the same number of single encryptions). The second and third subkeys M and N can be found as in the case of double encryption.

If the same plaintext is encrypted 2^{28} times using triple DES under 2^{28} different keys, an attacker can recover one of the 2^{28} keys using 2^{84} steps (and the same number of single encryptions). This result is due to Biham [1].

DESX is a variant of DES, where encrypting and decrypting requires to compute one single encryption and two XORs of s-bit blocks. Kilian and Rogaway [4] describe the security of DESX in the black-box-only model.

3 How to Save Steps

In this section, we describe an "operation optimized" attack to save some steps of computation, compared to MITM.

Consider a function $f : \{0,1\}^* \longrightarrow \{0,1\}^s$. A *collision* is a pair x, y of inputs with $x \neq y$ and $f(x) = f(y)$. The value $v \in \{0,1\}^s$ is associated with

[1] When computing the complexity of this stage, the operation of computing a value $b_{L,M} = E_M^2(E_L^1(p))$ *and* looking up the pair $(b_{L,M}, N)$ in a table *and* the operations to maintain the loop together count as *one step*.

a *t-collision*, if there exists a set S with $|S| \geq t$ inputs and $f(x) = v$ for all $x \in S$. Assuming the function $f : \{0,1\}^* \longrightarrow \{0,1\}^s$ to behave like a random function, and that $1 \leq w \leq 2^t$ inputs are randomly chosen, the expected number of values $v \in \{0,1\}^s$ associated with a t-collision[2] is about $w^t * 2^{-s(t-1)}$. Given plaintext/ciphertext pairs (p_i, c_i), our attack depends on finding t-collisions for the functions $f_{p_i} : \{0,1\}^k \longrightarrow \{0,1\}^s$, $f_{p_i}(L) = E^1_L(p_i)$. We consider all keys $L \in \{0,1\}^k$, hence the number of inputs for the function f_{p_i} is $w = 2^k$.

We write $K_1(a,i)$ for the set of all keys which encrypt the plaintext p_i to the ciphertext a using E^1. Similarly, we write $K_3(b,i)$ for all keys which decrypt the ciphertext c_i to b, using E^3. I.e.,

$$K_1(a,i) = \{\ L \in \{0,1\}^k \mid E^1_L(p_i) = a\ \} \quad \text{and}$$
$$K_3(b,i) = \{\ N \in \{0,1\}^k \mid E^3_L(b) = c_i\ \}.$$

If $|K_3(a,i)| \geq t$, the value a is associated with a t-collision. Given a pair (p_i, c_i), we choose set $S_A(i) \subseteq \{0,1\}^s$ of values associated with t-collisions:

$$S_A(i) = \{\ a \in \{0,1\}^s \mid \text{there exists a } t\text{-collision } K_3(a,i).\ \}.$$

Our attack works like this:

$i := 1$;
Repeat:
  let $(p_i, c_i) \in (\{0,1\}^s)^2$ be a known pair of plaintext and ciphertext;
  initialize the sets $K_1(\cdot, i)$, $K_3(\cdot, i)$, and $S_A(i)$ to be empty;
  A. for $L \in \{0,1\}^k$: $a := E^1_L(p_i)$;
    $K_1(a,i) := K_1(a,i) + \{L\}$;
    if $|K_1(a,i)| \geq t$ then $S_A(i) := S_A(i) + \{a\}$;
  ($*$ Now $S_A(i)$ is the set of all values a associated with a t-collision. $*$)
  for $N \in \{0,1\}^k$: $b := D^3_N(c_i)$;
    $K_3(b,i) := K_3(b,i) + \{N\}$;
  B. for $a \in S_A(i)$:
    for $M \in \{0,1\}^k$: $b := E^2_M(a)$;
      for $N \in K_3(b,i)$:
        for $L \in K_1(a,i)$: **tripletest**(i, L, M, N);
  $i := i + 1$;
until **tripletest** accepts.

The procedure "**tripletest**" can be realized like this:

Procedure **tripletest**$(i, L, M, N) =$
  $S_I := \{1, \ldots, l\} - \{i\}$;
  $d := 3k - s + \delta$;

[2] Cf. Rivest and Shamir [9], who exploit this for one of their micropayment schemes. To verify this estimation, one can use a well known special case, the "birthday paradox": *The expected number of inputs for f until the first 2-collision occurs is $c * 2^{s/2}$ with a small constant c.* (Actually, $c = \sqrt{\pi/2} \approx 1.25$, cf. [5, Section 2.1.5]).

repeat: choose $j \in S_I$ at random;
$S_I := S_I - \{j\}$;
$c := E_N^3(E_M^2(E_L^1(p_j)))$;
$d := d - s$;
until $(d \leq 0)$ or $(c_j \neq c)$;
if $(c_j = c)$ then accept (L, M, N) as the correct key-triple and stop
else reject (L, M, N) and continue.

When "`tripletest`" is called, the equation

$$E_N^3(E_M^2(E_L^1(p_i))) = c_i$$

holds. In the procedure, we are looking for $j \neq i$ such that $E_N^3(E_M^2(E_L^1(p_j))) \neq c_j$. If we fail often enough, i.e., $\lceil \frac{3k-s+\delta}{s} \rceil$ times, we accept the key-triple (L, M, N) as correct. The value δ serves as a security parameter, the risk to accept an incorrect key-triple is no more that $2^{-\delta}$.

On the average, a wrong key-triple requires insignificantly more than three single encryptions, i.e., one computation of $c := \ldots$, since for $j \in \{1, \ldots, l\} - \{i\}$, the equation $E_N^3(E_M^2(E_L^1(p_j))) = c_j$ holds one out of 2^s times. The correct triple is always accepted after $\lceil \frac{3k-s+\delta}{s} \rceil$ rounds. E.g., two rounds are sufficient for triple DES ($k = 56$ and $s = 64$) if $\delta = 20$. In the sequel, we assume δ to be "large enough" and ignore the risk of accepting an incorrect key-triple.

Let t be chosen such that $w^t * 2^{-s(t-1)} \ll 2^k$.

Theorem 1. *The expected number of pairs (p_i, c_i) for the operation optimized attack to succeed is $2^k/(w^t * 2^{-s(t-1)}t)$ (except for a small factor).*

Sketch of proof. Every t-collision $K_1(a, i) \in S_A(i)$ consists of at least t keys and hence has at least a $t * 2^{-k}$ chance to contain the correct first key L. Every index i corresponds to a pair (p_i, c_i) of plaintext and ciphertext. For every i, we expect to find about $w^t * 2^{-s(t-1)}$ values a to be associated with a t-collision $K_1(a, i)$. Thus the expected number of (plaintext,ciphertext)-pairs we need to consider in order to find the correct first key L is $2^k/(w^t * 2^{-s(t-1)}t)$.

It is easy to verify the following: If (L, M, N) is the correct key triple, $K_1(a, i) \in S_A(i)$ and $L \in K_1(a, i)$, then the procedure `tripletest`(i, L, M, N) is executed in stage B with the index i and the keys (L, M, N) as parameters. □

Theorem 2. *With l pairs (p_i, c_i), the operation optimized attack requires the following resources:*

- $\Theta(2^k)$ *units of storage space, and*
- $\Theta(w^t * 2^{-s(t-1)} * 2^k * l + w^t * 2^{-s(t-1)} * l * t * 2^{2k-s})$ *steps (and as much single encryptions).*

Proof. Both stages are to be executed l times. During every iteration of the "Repeat" loop, no results of previous iterations are needed. Hence, the amount of storage for the attack can be estimated by the storage space during one iteration, and the required number of steps is l times the average number of steps during

one iteration. Below, we estimate the storage space and the number of steps for one such iteration.

Both loops of stage A are iterated 2^k times, hence the number of steps is about $2*2^k$. When the first loop is finished, the storage space for the sets $K_1(a,i)$ (i.e., 2^k units) is no longer needed, and can be reused, with the exception of the sets $K_1(a,i) \in S_A(i)$. For the second loop, 2^k units of storages space are needed for the sets $K_3(\cdot,i)$. Since $w^t * 2^{-s(t-1)} \ll 2^k$, we expect the probability for $|S_A(i)| > 2^k$ to be negligible, and we approximate the storage space for stage A by 2^k.

Now, we consider stage B. For every pair (p_i, c_i) we expect the existence of $w^t * 2^{-s(t-1)}$ t-collisions; thus the loop "for $a \in S_A(i)$" is iterated $w^t * 2^{-s(t-1)}$ times on the average. The loop "for M ..." is iterated 2^k times, hence, $w^t * 2^{-s(t-1)} * 2^k$ single encryptions $b := E_M^2(a)$ are done. So far, we need $w^t * 2^{-s(t-1)} * 2^k$ steps. The expected size of a set $K_3(b,i)$ is $2^{k-s} \leq 1$. $K_1(a,i)$ is a t-collision, thus it contains about t keys L, and the procedure `tripletest` is to be called $w^t * 2^{-s(t-1)} * 2^k * 2^{k-s} * t$ times.

During each of the iterations of stage A and B and hence during the complete algorithm $\Theta(2^k)$ units of storage space are needed, and the number of steps is $l\Theta(2^k + w^t * 2^{-s(t-1)} * 2^k + w^t * 2^{-s(t-1)} * 2^k * 2^{k-s} * t) = \Theta(w^t * 2^{-s(t-1)} * 2^k + w^t * 2^{-s(t-1)} * 2^k * 2^{k-s} * t)$, similarly to the number of single encryptions. □

The constants hidden by the asymptotics are small. Given l pairs of plaintext and ciphertext, we need about 2^k units of storage space—as is the case for the MITM attack. The number of steps is $\text{STEPS}_A + \text{STEPS}_B^o + \text{STEPS}_B^i$. Here $\text{STEPS}_A \approx 2 * l * 2^k$ is the number of steps steps for stage A, $\text{STEPS}_B^o \approx l * w^t * 2^{-s(t-1)} * 2^k$ is the number of steps for the outer loops of stage B (i.e., the number of times the operation $b := E_M^2(a)$ is executed), and $\text{STEPS}_B^i \approx 3 * l * w^t * 2^{-s(t-1)} * 2^{2k-s}$ is the number of steps for all loops of stage B and for `tripletest`.

For triple DES ($k = 56$ and $s = 64$) expected number of t-collisions is about $2^{kt} * 2^{-s(t-1)}$. If $t = 8$, then $2^{kt} * 2^{-s(t-1)} = 2^0 = 1$, we expect one 8-collision, the attack requires about 2^{56} units of storage space, and for $l = 2^k/(w^t * 2^{-s(t-1)}t) = 2^{53}$ we need about $\text{STEPS}_A \approx 2^{110}$ steps (mainly table look-ups and single encryptions) for the attack—instead of 2^{111} similar steps for MITM.

We can improve this by choosing $t = 7$: The expected number of 7-collisions is $2^{kt} * 2^{-s(t-1)} = 2^8 = 256$. Again, the attack requires about 2^{56} units of storage space. For $l = 2^k/(w^t * 2^{-s(t-1)}t) = 2^{56}/(256 * 8) = 2^{45}$, we get $\text{STEPS}_A \approx 2^{102}$, $\text{STEPS}_B^i \approx 3 * 2^{104}$, and

$$\begin{aligned} \text{STEPS}_B^o &\approx l * w^t * 2^{-s(t-1)} * 2^k \\ &= (2^k/(w^t * 2^{-s(t-1)}t)) * w^t * 2^{-s(t-1)} * 2^k \\ &= 2^{2k}/7 = 2^{112}/7 \\ &\approx 2^{109.2}, \end{aligned}$$

thus we only need slightly more than 2^{109} steps (and as much single encryptions).

From a practical point of view, the operation optimized attack is not very useful for breaking triple DES. It is faster than MITM, but requires much more

pairs of known plaintexts/ciphertext (e.g., about 2^{45} if $t = 7$, compared to 3 for MITM). But from a theoretical point of view, the attack's performance clearly indicates triple DES to be weaker than widely believed.

4 How to Save Single Encryptions

The previous section's technique to reduce the number of steps seems to be at a dead end. So in the next two sections, we concentrate on reducing the number of single encryptions instead the number of steps. This section deals with an "encryption optimized" attack. Instead of l sets $S_A(i)$ depending on p_i, we choose one fixed set S_A and no longer exploit the occurances of t-collisions.

Let there be l plaintext/ciphertext pairs $(p_1, c_1), \ldots, (p_l, c_l)$ known to the attacker. In the previous section, we computed a set $S_A(i) \subseteq \{0,1\}^s$ for every index $i \in \{1, \ldots, l\}$. Now instead, we choose one set $S_A = S_A(1) = \cdots = S_A(l)$. The size $|S_A|$ of S_A is fixed and $|S_A| \ll 2^s$. We assume the $a \in S_A$ to be chosen randomly. (We use the independence of the sets S_A and $\{a \in \{0,1\}^s \mid a = E_L^1(p_i)\}$, where L denotes the correct first key.) Our attack consists of three stages:

1. For $a \in S_A$: compute the sets
$$S_1(a) = \left\{ (i, L) \in \{1, \ldots, l\} \times \{0,1\}^k \mid E_L^1(p_i) = a \right\}.$$
2. For $b \in \{0,1\}^s$ and $i \in \{1, \ldots, l\}$: compute the sets
$$K_3(b, i) = \left\{ N \in \{0,1\}^k \mid E_N^3(b) = c_i \right\}.$$
3. For $M \in \{0,1\}^k$ and $a \in S_A$:
   $b := E_M^2(a)$;
   for $(i, L) \in S_1(a)$:
   for $N \in K_3(b, i)$:
   `tripletest`(i, L, M, N).

What is the chance to find the correct key by using the algorithm?

Theorem 3. *If $|S_A| \approx 2^s/l$, the encryption optimized attack's probability to find the correct key-triple is close to* $1/2$.

Sketch of proof. The attack succeeds in finding the correct key triple (L, M, N), if for any $i \in \{1, \ldots, l\}$ the operation "`tripletest`(i, L, M, N)" is executed, i.e., if a pair (i, a) exists in $\{1, \ldots, l\} \times S_A$ with $E_L^1(p_i) = a$. The existence of such a pair can be expected if $l * |S_A| \approx 2^s$ (due to the birthday paradox). □

One of the resources required to mount the attack is the number l of known plaintext/ciphertext pairs. What about the other resources?

Theorem 4. *If $|S_A| \approx 2^s/l$, $s \geq k$ and $2^{2k-s} \geq l \geq 2^{2s-2k}$, the encryption optimized attack requires the following resources:*

- *Exactly l known plaintext/ciphertext pairs,*

– $\Theta(l * 2^k)$ *units of storage space,*
– $\Theta(2^{2k})$ *steps,*
– *and* $\Theta(2^{3k-s})$ *single encryptions.*

Proof. Let the sets $S_1(\cdot)$ and $K_3(\cdot,\cdot)$ be initialized to be empty. The first two stages can be realized like this:

1. For $i \in \{1,\ldots,l\}$ and $L \in \{0,1\}^k$: $a := E_L^1(p_i)$;
   $S_1(a) := S_1(a) + \{(i,L)\}$.
   (* Or: If $a \in S_A$, then $S_1(a) := \ldots$ *)
2. For $i \in \{1,\ldots,l\}$ and $N \in \{0,1\}^k$: $b := D_N^3(c_i)$;
   $K_3(b,i) := K_3(b,i) + \{N\}$.

Each loop is iterated $l * 2^k$ times, hence the overall number of elements in the sets $K_3(\cdot,\cdot)$ are $l * 2^k$, the overall number of elements in the sets $S_1(\cdot)$ is the same (but we only need the sets $S_1(a)$ for $a \in S_A$), and the number of steps and single encryptions for the first two stages is $\Theta(l * 2^k)$.

Stage 3 requires much less storage than the first two stages. Its outer loop "For $M \in \{0,1\}^k$ and $a \in S_A$" is iterated $2^k * |S_A| \approx 2^{k+s}/l$ times. On the average, the middle loop "for $(i,L) \in S_1(a)$" is iterated $l * 2^k/2^s$ times, and the inner loop "for $N \in K_3(b,i)$" is iterated 2^{k-s} times. Since $2^{k-s} < 1$, the outer and the middle loop determine the number $(2^k * |S_A|)(l * 2^k/2^s) \approx 2^{2k}$ of steps for stage 3. But for the single encryptions, we count how often the operation "$b := E_M^2(a)$" is executed in the outer loop (i.e., $2^k * |S_A| \approx 2^{k+s}/l$) and add three times the number the operation "$c := E_N^3(E_M^2(E_L^1(p_j)))$" is executed within the procedure `tripletest` (i.e., $3(2^k * |S_A|)(l * 2^k/2^s)(2^{k-s}) \approx 3 * 2^{3k-s}$). If $l \geq 2^{2s-2k}$, as required, the `tripletest` part dominates the sum, i.e., the number of single encryptions in stage 3 is about $3 * 2^{3k-s} = \Theta(2^{3k-s})$.

Thus the storage requirement for the attack is dominated by stage 2, the number of steps and the number of single encryptions are dominated by stage 3. Hence we need $\Theta(l * 2^k)$ units of storage space, $\Theta(2^{2k})$ steps and especially $\Theta(2^{3k-s})$ single encryptions. □

As one can easily deduce from the proof, the constants hidden by the asymptotics are small. We need about $l * 2^k$ units of storage, about 2^{2k} steps, and about $3 * 2^{3k-s} + 2^{k+s}/l$ single encryptions. For triple DES, we may choose l within the range $2^{16} \leq l \leq 2^{48}$. Given, say, $l = 2^{16}$ known pairs of plaintext and ciphertext, we need roughly 2^{72} units of storage (mainly for the elements of the sets $K_3(\cdot,\cdot)$), cycle through a loop for about 2^{112} times, and have to encrypt/decrypt about

$$\begin{aligned} & 3 * 2^{3k-s} + 2^{k+s}/l + l * 2^{k+1} \\ \approx\ & 3 * 2^{104} + 2^{104} + 2^{73} \\ \approx\ & 2^{106} \end{aligned}$$

times.

Comparing this section's encryption optimized attack with the operation optimized one, we find to have reduced the number of single encryptions, but not the number of steps.

5 How to Save more Single Encryptions

The encryption optimized attack's efficiency is limited, since `tripletest` is executed 2^{3k-s} times, which induces $3 * 2^{3k-s}$ single encryptions. As we argued in the sketch of proof of theorem 3, the correct key triple (L, M, N) is found "if a pair (i, a) exists in $\{1, \ldots, l\} \times S_A$ with $E_L^1(p_i) = a$." In this section, we modify the attack; we only execute `tripletest` if there exist *two* pairs $(i, a), (j, a') \in \{1, \ldots, l\} \times S_A$ with $E_L^1(p_i) = a$ and $E_L^1(p_j) = a'$. This idea leads to the "advanced attack". (More generally, we execute `tripletest` if r pairs $(i_1, a_1), \ldots, (i_r, a_r)$ with $E_L^1(p_j) = a_j$ exist in $\{1, \ldots, l\} \times S_A$. In this paper, we concentrate on $r \in \{1, 2\}$.) On one hand, this forces us to increase the number of known plaintext/ciphertext pairs $(p_\cdot, c_\cdot)$ in order to succeed. On the other hand, we need to execute the `tripletest` much less frequently.

The first two stages are the same as before, for stage 3 we do the following:

3. For $M \in \{0,1\}^k$:
   $S := \{\}$;
   for $a \in S_A$:
   $b := E_M^2(a)$;
   for $(i, L) \in S_1(a)$:
   for $N \in K_3(b, i)$:
   if $(L, N) \in S$ then `tripletest`(i, L, M, N)
   else $S := S + \{(L, N)\}$.

Theorem 5. *If $|S_A| \approx 2 * 2^s/l$, the advanced attack's probability to find the correct key-triple is close to $1/2$.*

Sketch of proof. Let (L, M^*, N) denote the correct key triple. We consider the iteration of the loop "For $M \in \{0,1\}^k$:" with $M = M^*$, all other iterations cannot succeed anyway. If $|S_A| \approx 2 * 2^s/l$, the expected number r of pairs $(i_1, a_1), \ldots, (i_r, a_r) \in \{1, \ldots, l\} \times S_A$ with $E_L^1(p_j) = a_j$ is $r = 2$. If there actually exist two such pairs (i_1, a_1) and (i_2, a_2) in $\{1, \ldots, l\} \times S_A$, then the following inclusions hold

$$\begin{array}{lll} (i_1, L) \in S_1(a_1), & & N \in K_3(E_M^2(a_1), i_1), \\ (i_2, L) \in S_1(a_2), & \text{and} & N \in K_3(E_M^2(a_1), i_2). \end{array}$$

In this case, the key pair (L, N) is found twice within the execution of the algorithm. At first "$(L, N) \in S$" is wrong and (L, N) is inserted into the set S. The second time "$(L, N) \in S$" is true, `tripletest`(i, L, M, N) is executed (with $i \in \{i_1, i_2\}$) and accepts because $(L, M, N) = (L, M^*, N)$ is the correct key triple. □

Theorem 6. *If $|S_A| \approx 2^{s+1}/l$, the advanced attack requires the following resources:*

- *Exactly l known plaintext/ciphertext pairs,*
- *$\Theta(l * 2^k)$ units of storage space,*

– $\Theta(2^{2k})$ *steps,*
– *and* $\Theta(l * 2^k + 2^{k+s}/l)$ *single encryptions.*

Proof. The resource requirements for the first two stages of the advanced attack are the same as for the encryption optimized attack.

In the third stage, and for fixed M and a, the loop "for $(i, L) \in S_1(a)$" is iterated about $l * 2^{k-s}$ times, the inner loop "for $(L, N) \in S$" is iterated about 2^{k-s} times. Hence the size of the set S is roughly $|S| \approx l * 2^{2k-2s} \leq l$. The sets $K_3(\cdot, \cdot)$ require $l \times 2^k = \Theta(l * 2^k)$ units of storage and thus dominate the advanced attack's storage requirements.

Similarly to the proof of theorem 4, the number of steps is $(2^k * |S_A|)(l * 2^k/2^s) \approx 2^{2k+1} = \Theta(2^{2k})$.

The first two stages together require $l*2^{k+1}$ single encryptions. The operation $c := E_N^3(\ldots)$ in the procedure `tripletest` is to be executed about $2^{3k-2s+1}$ times, inducing $3 * 2^{3k-2s+1}$ single encryptions. The operation $b := E_M^2(a)$ is to be executed $2^k * |S_A| \approx 2^{k+s+1}/l$ times. Since $l * 2^{k+1} \gg 3 * 2^{3k-2s+1}$, the number of single encryptions is about

$$l * 2^{k+1} + 3 * 2^{3k-2s+1} + 2^{k+s+1}/l \;\approx\; l * 2^{k+1} + 2^{k+s+1}/l,$$

i.e., $\Theta(l * 2^k + 2^{k+s}/l)$. □

In practice, we need about $l * 2^k$ units of storage, about 2^{2k+1} steps, and about $2^{k+s+1}/l + l * 2^{k+1}$ encryptions/decryptions. If we fix $l = 2^{s/2}$, we need

$$\text{about} \quad 2^{k+(s/2)+2} \quad \text{single encryptions.} \tag{1}$$

For attacking triple DES, given $l = 2^{32}$ known pairs of plaintext and ciphertext, we need 2^{88} units of storage space and 2^{113} steps, but only 2^{90} single encryptions. In comparison to the operation optimized attack, the advanced attack allows us to drastically reduce the amount of single encryptions at the cost of doubling the number of steps. So what is our gain? As we mentioned in the introduction, a single encryption is a very complex operation, compared to, say, table look-ups. If we assume one implementation of DES to require 8 table look-ups per round, i.e., $8 * 16 = 2^7$ table look-ups per encryption, our speed-up can be estimated like this:

– The expected number of 2^{111} steps and as much single encryptions of the MITM attack actually correspond to about $1.3 * 2^{109}$ triple encryptions.
– The operation optimized attack of section 3 needed 2^{109} steps and single encryptions. These correspond to about $1.3 * 2^{107}$ triple encryptions.
– The encryption optimized attack's 2^{112} steps (mostly table look-ups) and 2^{106} single encryptions. This is equivalent to about 2^{105} triple encryptions.
– This section's attack requires 2^{113} steps (mostly table look-ups) and 2^{90} single encryptions. This corresponds to about $1.3 * 2^{104}$ triple encryptions.

In the remainder of this section, we concentrate on the black-box-only model. Kilian and Rogaway [4] consider the DESX block cipher and its security. A generalized variant is EX, based on the encryption function $E : \{0,1\}^k \times \{0,1\}^s \longrightarrow$

$\{0,1\}^s$. An EX key is a triple $(L, M, N) \in \{0,1\}^k \times \{0,1\}^s \times \{0,1\}^s$. The encryption function is $EX_{(L,M,N)}(p) = N \oplus E_L(M \oplus p)$, where "$\oplus$" denotes the bit-wise XOR. Compared to triple DES, DESX is amazingly elegant and efficient.

Let l denote the number of known (or chosen) pairs of plaintext and ciphertext. Kilian and Rogaway prove for EX that the attacker's advantage in distinguishing between random nonsense unrelated to E, and EX encryptions using a key-triple (L, M, N) unknown to the attacker, is $\epsilon \leq l * x * 2^{-k-s+1}$. Here, x denotes the number of single encryptions. If $\epsilon = 1/2$ and $l = s/2$, this requires

$$\text{about} \quad x \geq 2^{k+(s/2)-2} \quad \text{single encryptions,} \tag{2}$$

e.g., about $x \geq 2^{85}$ for DESX. (Note that Kilian and Rogaway consider $k = 55$ and ignore the additional key bit of DES. This is necessary for lower bounds in the black-box-only model due to the DES complementation property.) By presenting a chosen plaintext attack, Kilian and Rogaway also demonstrate that the above bound is tight, except for a small factor.

Note that our result (1) for breaking triple encryption is close to Kilian's and Rogaway's lower bound (2) for EX. We conclude, in order to find a combined cipher *provably much more secure* than EX (or DESX), one has to abstain from triple encryption (triple DES) or to forego the black-box-only model. A model more accurate than the black-box-only model is likely to be more complicated, too.

6 A Special Variant for Triple DES

So far, we pretended the underlying single block cipher to be ideal, i.e., to behave like a random permutation. But DES is not an ideal block cipher. Most important in this context is the complementation property: If $\overline{x}$ denotes the complement of the bit-string x, then for every plaintext $p \in \{0,1\}^s$ and every key $K \in \{0,1\}^k$:

$$\mathrm{DES}_K(p) \;=\; \overline{\mathrm{DES}_{\overline{K}}(\overline{p})}.$$

How does the complementation property affect the efficiency of our attacks?

First, we note there is not much harm for the attacker. The encryption optimized attack succeeds, if the sets $\{p_1, \ldots, p_l\}$ and S_A are chosen such that there exists a $(i, a) \in \{1, \ldots, l\} \times S_A$ with $E^1_L(p_i) = a$, L the correct first subkey, cf. proof of theorem 3. This probability is not at all affected by the complementation property $E^1_{\overline{L}}(\overline{p_i}) = \overline{a}$. We may argue similarly for the advanced attack. The success rate of the operation optimized attack depends on the probability that for a plaintext p_i the correct first subkey L participates in a t-collision $K(a, i) = \{L, L_2, \ldots, L_t\}$, i.e., $E^1_L(p_i) = E^1_{L_2}(p_i) = \ldots = E^1_{L_t}(p_i) = a$. Again, this probability is not affected by the complementation property $E^1_{\overline{L}}(\overline{p_i}) = \overline{a}$.

Second, there are many ways for the attacker to exploit the complementation property for a small improvement of an attack. For the sake of shortness, we concentrate on one example. Recall the attack in section 3. Let S_A be chosen such that for all $a \in \{0,1\}^s$ the equivalence $a \in S_A \iff \overline{a} \in S_A$ holds. The attack is unchanged, except for stage B:

B. for $a \in S_A(i)$:
    for $M \in \{0\} \times \{0,1\}^{k-1}$:
        $b := E^2_M(a)$;
        for $N \in K_3(b,i)$:
            for $L \in K_1(a,i)$: `tripletest`(i,L,M,N);
        (* Next, we exploit $\overline{b} = E^2_{\overline{M}}(\overline{a})$. *)
        for $N \in K_3(\overline{b},i)$:
            for $L \in K_1(\overline{a},i)$: `tripletest`$(i,L,\overline{M},N)$;

The analysis in section 3 is not much affected. Neither the expected number of pairs of plaintext and ciphertext changes, nor the complexity STEPS_A of stage A, nor the attack's storage requirements.

With respect to stage B, the loop "for $a \in S_A(i)$" is iterated $w^t * 2^{-s(t-1)}$ times on the average. The loop "for $M \ldots$" is only iterated 2^{k-1} times, hence, $w^t * 2^{-s(t-1)} * 2^{k-1}$ single encryptions $b := E^2_M(a)$ are done. So far, we need $\text{STEPS}^o_B = w^t * 2^{-s(t-1)} 2^{k-1}$ steps for stage B. Together, the two loops "for $N \in \ldots$" need as much time as before: $\text{STEPS}^i_B = w^t * 2^{-s(t-1)} * 2^k * 2^{k-s} * t$.

If we choose the parameters were $t = 7$, $u \approx 2.2$, and $l = 2^{45}$, the operation optimized attack's complexity is the sum of three numbers $\text{STEPS}_A \approx 2^{102}$, $\text{STEPS}^i_B \approx 3 * 2^{104}$, and $\text{STEPS}^o_B \approx 2^{109.2}$.

This section's variant does not affect STEPS_A and STEPS^i_B, hence

$$\text{STEPS}^o_B \approx 2^{108.2}$$

approximates the overall number of steps and single encryptions.

7 Comparison and Conclusion

Based on today's technology, neither MITM nor any of our attacks constitutes a practical way to break triple DES. If in the future an attack like MITM will be considered practical for doing this, certainly some of the required resources will be more valuable than others. This paper provides a variety of options how to possibly save such a bottleneck resource. A comparison is given in table 1.

Van Oorschot and Wiener [7,8] considered attacks with decreased memory requirements at the cost of increased running times. Usually, reducing storage requirements is seen as the main goal of improving an attack like MITM. The approach in sections 4 and 5 is to decrease the running time at the cost of storage. As an anonymous referee criticized, this seems to make our attacks less realistic. The current author's reply is that the basic MITM attacks on double encryption and two-key triple encryption both have balanced time-memory characteristics, i.e., require roughly one step of computation per unit of memory. In this case, trading away storage space at the cost of additional computational steps, as van Oorschot and Wiener did, certainly makes such attacks more realistic. On the other hand, the MITM attack on general (three-key) triple encryption has

attack	sect.	l	memory	steps	single encryptions
MITM	2	3	2^{56}	2^{111}	2^{111}
op. optim.	3	2^{45}	2^{56}	$2^{109.2}$	$2^{109.2}$
(variant)	6	(2^{45})	2^{56}	$2^{108.2}$	$2^{108.2}$
encr. optim.	4	l	$l * 2^k$	2^{2k}	$3 * 2^{3k-s} + 2^{k+s}/l$
		2^{16}	2^{72}	2^{112}	2^{106}
		2^{24}	2^{80}	2^{112}	$3 * 2^{104}$
		2^{32}	2^{88}	2^{112}	$3 * 2^{104}$
advanced	5	l	$l * 2^k$	2^{2k+1}	$l * 2^{k+1} + 2^{k+s+1}/l$
		2^{16}	2^{72}	2^{113}	2^{105}
		2^{24}	2^{80}	2^{113}	2^{97}
		2^{32}	2^{88}	2^{113}	2^{90}

Table 1. Attacking triple DES with l known (chosen) pairs of plaintext and ciphertext and the expected number of resources required.

a highly unbalanced time-memory characteristic: 2^k units of memory and 2^{2k} steps are needed, i.e., 2^k steps per unit of memory. If k is reasonably large, e.g., $k = 56$, decreasing the running time at the cost of additional memory requirements actually appears to make such attacks *more* realistic. (Today though, our attacks are far from being practical, as is the MITM attack. It is quite difficult to reasonably estimate the economically best time-memory characteristic of a future technology for which such attacks are practical.)

Even though our attacks are far from being practical today, this paper demonstrates that *it is too optimistic to identify the complexity of breaking triple DES and similar block ciphers with the complexity of the MITM attack.* Also, this paper alludes that the ability to quickly perform many single DES operations is not crucial for breaking triple DES (though even the required number of single DES operations is too large to be considered feasible today). The number of memory accesses, i.e., table look-ups, appears to be dominating—with great consequences on the difficulty of massively parallel triple DES cracking. Finally, this paper gives evidence that it will be difficult to prove triple DES to be much stronger than the more efficient DESX.

8 Acknowledgements

The author is thankful to Rüdiger Weis for discussing DESX and very much appreciates Eli Biham's and the anonymous referees' aid in improving the presentation of this material.

References

1. E. Biham, "How to forge DES-Encrypted Messages in 2^{28} steps", Technical report CS0884, Computer Science department, Technion, 1996, found in the www[3], last checked at Dec 17, 1997.
2. J. Kelsey, B. Schneier, D. Wagner, "Key-Schedule Cryptanalysis of 3-WAY, IDEA, G-DES, RC4, SAFER, and Triple-DES", *Crypto '96*, Springer LNCS 1109, 237–251.
3. J. Kilian, P. Rogaway, *How to protect DES against exhaustive key search*, *Crypto '96*, Springer LNCS 1109, 252–267.
4. J. Kilian, P. Rogaway, *How to protect DES against exhaustive key search* (full version of [3]), found in the www[4], last checked: Dec 17, 1997.
5. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.
6. R.C. Merkle, M.E. Hellman, "On the security of multiple encryption" *Communications of the ACM*, Vol. 24, No. 7 (1981).
7. P.C. van Oorschot, M.J. Wiener, "A known-plaintext attack on two-key triple encryption", *Eurocrypt '90*, Springer LNCS 473, 318–325.
8. P.C. van Oorschot, M.J. Wiener, "Improving implementable meet-in-the-middle attacks by orders of magnitude", *Crypto '96*, Springer LNCS 1109, 229–236.
9. R.L. Rivest, A. Shamir, "PayWord and MicroMint, two simple micropayment schemes", *CryptoBytes*, Vol. 2, No. 1 (1996), 7–11.

This paper was published at:
S. Vaudenay (ed.): Fast Software Encryption (1998), Springer LNCS.

[3] `http://www.cs.technion.ac.il/Reports/`
[4] `http://wwwcsif.cs.ucdavis.edu/~rogaway/papers/list.html`