

RELATED-KEY ATTACK AGAINST TRIPLE ENCRYPTION BASED ON FIXED POINTS

Serge Vaudenay

EPFL

CH-1015 Lausanne, Switzerland
<http://lasecwww.epfl.ch>

Keywords: Triple-encryption, Cryptanalysis

Abstract: Triple encryption was proposed to increase the security of single encryption when the key is too short. In the past, there have been several attacks in this encryption mode. When triple encryption is based on two keys, Merkle and Hellman proposed a subtle meet-in-the-middle attack which can break it at a price similar to breaking single encryption (but with nearly all the code book). When triple encryption is based on three keys, Kelsey, Schneier, and Wagner proposed a related-key attack which can break it at a price similar to breaking single encryption.

In this paper, we propose a new related-key attack against triple encryption which compares to breaking single encryption in the two cases. Our attack against two-key triple-encryption has exactly the same performances as a meet-in-the-middle on double-encryption. It is based on the discovery of fixed points in a decrypt-encrypt sequence using related keys.

In the two-key case, it is comparable to the Merkle-Hellman attack (except that it uses related keys). In the three-key case, it has a higher complexity than the Kelsey-Schneier-Wagner attack but can live with known plaintexts.

1 INTRODUCTION

A classical security model for symmetric encryption is the key recovery under chosen plaintext or ciphertext attacks. Since ciphers are broken by generic attacks such as exhaustive search, we must live with these attacks and hope that their complexity is the minimal cost for breaking the cipher. Indeed, a cipher is secure if there is no attack better than exhaustive search, i.e. if its complexity is lower than 2^ℓ where ℓ is the key length.

In the 90's Biham and Knudsen proposed the notion of related-key attack in which an adversary can impose to change the secret key following some chosen relation ϕ (Biham, 1993; Biham, 1994; Knudsen, 1992). One problem is that related-key attacks open the way to new generic attacks such as the ones by Biham (Biham, 1996). So, exhaustive search may no longer be the reference for assessing the security of a cipher.

As an example of related-key attack, Kelsey, Schneier, and Wagner presented an attack against three-key triple encryption which shows that this is not more secure than single encryption (Kelsey,

Schneier, Wagner, 1996).

In this paper, we first discuss on various ways to compare the complexity related-key attacks. Following a full-cost model, an attack is significant $\frac{tm}{p} < 2^\ell$, where r is the number of related keys, t (resp. m) is the time (resp. memory) complexity, and p is the probability of success. In a more conservative approach, we shall compare $\max\left(\frac{t}{p}, m\right)$ with $2^{\frac{\ell}{2}}$. We can also consider comparison in a restricted attack model in order to limit some characteristics such as the number of related keys. Then, we provide a related-key attack against two-key triple encryption.

Related work on triple-DES. As far as we know, the only related-key attacks against triple-DES are the generic attack from Biham, the Kelsey-Schneier-Wagner attack, and an attack from Phan (Biham, 1996; Kelsey, Schneier, Wagner, 1996; Phan, 2004). There are other attacks using no related keys such as attacks based on meet-in-the-middle by Merkle and Hellman, known plaintext variants by Van Oorschot and Wiener, and a nice optimization by Lucks (Merkle and Hellman, 1981; Lucks, 1998;

van Oorschot and Wiener, 1990; van Oorschot and Wiener, 1999). We use the table from Phan to compare these attacks with ours (Phan, 2004). The figures¹ are given on Table 1. Previous attacks are discussed in this paper.

Our contribution. In this paper, we present a new attack on triple encryption which is based on the discovery of fixed points for the mapping

$$x \mapsto \text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1}$$

for some relation φ . This discovery requires the entire code book in a *Broadcast Known Plaintext (BKP) attack* for Enc_K and $\text{Enc}_{\varphi(K)}$ which makes our data complexity high. In the BKP model, the adversary obtains a random plaintext and its encryption under different keys. Once we have a (good) fixed point, our attack becomes similar to a standard meet-in-the-middle attack. So, it has a pretty low complexity. Finally, we show that our attack compares well to the best ones so far. In the 2-key case, it becomes the best known-plaintext attack.

2 COMPARING RELATED-KEY ATTACKS

Given a dedicated attack against a cipher, it is tempting to compare it with exhaustive search and declare the cipher broken if the attack is more efficient. This is however a bit unfair because the attack model may already have better generic attacks than exhaustive search.

As an example, Biham’s generic attack (Biham, 1996) applies standard time-memory tradeoffs in the related key model. His attack consists of collecting $y_i = \text{Enc}_{K_i}(x)$ for a fixed x and r related keys. That is, we use r chosen plaintexts. Then, it builds a dictionary (y_i, i) and run a multi-target key recovery to find one K such that $\text{Enc}_K(x)$ is in the dictionary. With t attempts, the probability of success is $p = 1 - (1 - r2^{-\ell})^t \approx 1 - e^{-rt2^{-\ell}}$. The dictionary has size $m = r(\ell + \log r)$ bits. For simplicity, we approximate $m \approx r$. In particular, for $t = r = 2^{\ell/2}$, we have $p \approx 1 - e^{-1} \approx 63\%$, so this is much cheaper than exhaustive search.

The complexity of a related-key attack can be characterized by a multi-dimensional vector consisting of

¹with slightly different units: our time complexities are measured in terms of triple encryption instead of single encryption; our memory complexities are measured in bits instead of 32-bit words; our number of keys include the target one and not only the related ones

- the number of related keys r (the number of keys which are involved is r , i.e. $r = 1$ when the attack uses no related keys);
- the data complexity d (e.g. the number of chosen plaintexts), where we may distinguish known plaintexts (KP), broadcast known plaintexts (BKP), chosen plaintexts (CP), and chosen ciphertexts (CC) as they may be subject to different costs in the attack model;
- the time complexity of the adversary t , where we may distinguish the precomputation complexity and the online running time complexity;
- the memory complexity m , which may further distinguish quick-access or slow-access memory, read/write memory or read-only memory;
- the probability of success p .

There are many other possible refinements.

We can compare attacks by using the partial ordering \leq_p on vectors $(r, d, t, m, \frac{1}{p})$, i.e.

$$(r, d, t, m, p) \leq_p (r', d', t', m', p')$$

$$\Downarrow$$

$$r \leq r' \text{ and } d \leq d' \text{ and } t \leq t' \text{ and } m \leq m' \text{ and } p \geq p'$$

When a category such as the data complexity d has a sub-characterization $(d_{KP}, d_{BKP}, d_{CP}, d_{CC})$, the $d \leq d'$ implies an other partial ordering on these sub-characteristics. We can say that an attack is *insignificant* if there is a generic attack with a lower complexity vector. It is not always possible to compare two multi-dimensional vectors. So, it is not clear whether an attack is significant when it is not insignificant. So, it is quite common to extend the partial ordering \leq_p using different models which are discussed below.

Conservative model. Traditionally, t , m , and p are combined into a “complexity” which is arbitrarily measured by $\max\left(\frac{t}{p}, m\right)$. We could equivalently adopt $\frac{t}{p} + m$ since these operations yield the same orders of magnitude.

The idea behind this arbitrary notion is that we can normalize the success probability p by using $\frac{1}{p}$ sessions of the attack. So, t has a factor $\frac{1}{p}$ corresponding to $\frac{1}{p}$ different sessions. Clearly, the running time of every session adds up whereas their memory complexity does not. If we make no special treatment for r and d , we can just extend this simple notion by adding them in the time complexity t (since the adversary must at least read the received data). We can thus replace t by $\max(r, d, t)$. This leads us to

$$C_{\text{conservative}}(r, d, t, m, p) = \max\left(\frac{r}{p}, \frac{d}{p}, \frac{t}{p}, m\right)$$

Table 1: Attacks against Triple-DES.

target	data	parameters			complexity	reference
		memory	time	#keys	$C_{\text{conservative}}$	
2K-3DES	2 (KP)	2^7	2^{112}	1	2^{112}	exhaustive search (Biham, 1996) (generic)
	2^{56} (CP)	2^{62}	2^{56}	2^{56}	2^{62}	
	2^{56} (KP)	2^{58}	2^{112}	1	2^{112}	(van Oorschot and Wiener, 1990; van Oorschot and Wiener, 1999)
	2^{33} (KP)	$2^{91.5}$	2^{86}	2	$2^{91.5}$	(Phan, 2004)
	2^{65} (KP)	2^{72}	2^{56}	2	2^{72}	this paper
	2^{65} (BKP)	2^{63}	2^{56}	2	2^{66}	this paper
	2^{64} (KP)	2^{64}	2^{56}	1	2^{64}	(Merkle and Hellman, 1981) variant
3K-3DES	2^{56} (CP)	2^{63}	2^{56}	1	2^{63}	(Merkle and Hellman, 1981)
	3 (KP)	2^8	2^{168}	1	2^{168}	exhaustive search (Biham, 1996) (generic)
	2^{84} (CP)	2^{92}	2^{84}	2^{84}	2^{92}	
	2^{32} (KP)	2^{90}	2^{104}	1	2^{104}	(Lucks, 1998)
	3 (CP)	2^{58}	2^{110}	1	2^{110}	(Merkle and Hellman, 1981)
	2^{33} (KP)	2^{35}	2^{86}	2	2^{86}	(Phan, 2004)
	2^{67} (KP)	2^{72}	2^{57}	6	2^{72}	this paper
	2^{67} (BKP)	2^{63}	2^{57}	6	2^{67}	this paper
	2 (BKP)	2^{58}	2^{54}	2	2^{58}	(Kelsey, Schneier, Wagner, 1996)

For instance, the Biham attacks (Biham, 1996) have complexity $2^{\frac{1}{2}\ell}$.²

In some cases, there may be a special treatment for r and d though, especially regarding the $\frac{1}{p}$ factor. Actually, the current $\frac{1}{p}$ factor corresponds to the worst case where iterating an attack requires new related keys. In many cases, related keys could just be reused, which means that the total number of related keys may be r instead of $\frac{r}{p}$. We can just keep in mind that the $C_{\text{conservative}}$ formula may not be well adapted to attacks with a probability of success far from 1. We should rather normalize the attack using the most appropriate technique before applying the formula on the normalized attack.

This kind of rule of the thumb is pretty convenient because two attacks can always be compared by $C_{\text{conservative}}$: let

$$(r, d, t, m, p) \leq_{\text{conservative}} (r', d', t', m', p')$$

$$\Downarrow$$

$$C_{\text{conservative}}(r, d, t, m, p) \leq C_{\text{conservative}}(r', d', t', m', p')$$

This defines a total ordering. An attack is said conservative-significant if it is better than generic ones following the conservative ordering. That is, an attack is conservative-significant if and only if

$$C_{\text{conservative}}(r, d, t, m, p) < 2^{\frac{\ell}{2}}$$

²Strictly speaking, we shall have a $\frac{1}{p}$ factor corresponding to $p = 63\%$ but this would give the same order of magnitude and this simple formula aims at comparing orders of magnitude.

Limited related-key models. Arguably, related keys (or event chosen plaintexts or ciphertexts) are harder to obtain, compared to spending time in the attack. Namely, an attack of complexity $2^{\frac{3}{4}\ell}$, $r = 1$, and $d = 1$ is declared not significant with $\leq_{\text{conservative}}$ because of the Biham attack (Biham, 1996) of complexity $2^{\frac{1}{2}\ell}$, which is a bit unfair. So, we could either go back to some partial ordering or to some attack model restrictions. For instance, a common model (when we do not care about related-key attacks) consists of limiting to $r = 1$. A natural model would consist of limiting $r \leq B_r$ for some bound B_r . Finally, we can compare an attack with the best generic one using no more related keys. That is, we say that an attack is *conservative-significant in the RK-limited model* if its conservative complexity is better than the one for all generic attacks using no more related keys. If (t, r, d, m, p) is the complexity vector of the attack, we shall compare $C_{\text{conservative}}(r, d, t, m, p)$ with the one of all $(t', r', r', r', 1 - e^{-r't/2^{-t}})$ for all t' and $r' \leq r$. Clearly, the minimal complexity is reached for $r' = r$ and $t' = 2^\ell / r'$. So, the attack is conservative-significant in the RK-limited model if

$$C_{\text{conservative}}(r, d, t, m, p) < \frac{2^\ell}{r}$$

Other limited models. We may also consider other limited models. For instance we can restrict to attacks using known plaintexts only. All combinations of limitations can be imagined. The relevance of these

limited models shall be driven by significance for applications.

Full-cost model. Wiener introduced the full cost expressed as $O(t + \frac{tm}{c} + t\sqrt{cp^3})$ where c is the number of processors and p is the rate of access to the memory of all processors per time unit (Wiener, 2004). (We assume here that parameters are normalized so that we can assume $p = 1$.) Using a single processor and $p = 1$, this simplifies to $O(tm)$. Again, we replace t by $\max(r, d, t)$ to integrate r and d . So, we could also consider

$$C_{\text{full}}(r, d, t, m, p) = \max(r, d, t) \frac{m}{p} \quad (1)$$

and define

$$\begin{aligned} (r, d, t, m, p) &\leq_{\text{full}} (r', d', t', m', p') \\ &\quad \updownarrow \\ C_{\text{full}}(r, d, t, m, p) &\leq C_{\text{full}}(r', d', t', m', p') \end{aligned}$$

The total ordering which takes parallelism tricks is a bit more complicated. Without using any parallelism trick, Biham's generic attacks have $d = r$, $t = 2^\ell/r$, and $m = r$. So, their full cost is $\max(r^2, 2^\ell)$. Again, this is relevant for $r \leq 2^{\frac{\ell}{2}}$ only and the full cost is 2^ℓ no matter r . In this case, exhaustive search with $r = 2^\ell$ has the same full cost. An attack is full-significant if and only if

$$C_{\text{full}}(r, d, t, m, p) < 2^\ell$$

As a rule of the thumb, we could adopt the simple criterion $\frac{tm}{p} < 2^\ell$.

Note that Equation (1) only gives an upper bound on the full cost which can be pessimistic. For instance, it was shown that meet-in-the-middle (Diffie and Hellman, 1977) with a key of size ℓ_k has a full cost of $2^{\frac{4}{3}\ell_k}$ and may also be reduced to $2^{\frac{6}{5}\ell_k}$ using parallelism (Wiener, 2004). So, the comparison based on full cost shall be done with great care.

As an application we can look at recent attacks on AES working with $p = 1$. (See Table 2.) As we can see, the Biryukov-Khovratovich attack (Biryukov and Khovratovich, 2009) on AES-192 is only conservative-significant in the RK-limited model, thanks to the low number of related keys, but it is not conservative-significant. The Biryukov-Khovratovich-Nikolić attack (Biryukov, Khovratovich, Nikolić, 2009) on AES-256 is conservative-significant in the RK-limited model, thanks to the low number of related keys, but it is not conservative-significant. The Biryukov-Khovratovich attack (Biryukov and Khovratovich, 2009) on AES-256 is significant for all criteria.

3 SEMI-GENERIC RELATED-KEY ATTACKS AGAINST 3DES

We propose here a related-key attack against 3DES. It is semi-generic in the sense that it does not depend on DES but only on the structure of triple encryption which is used in 3DES, that is the encrypt-decrypt-encrypt structure. We consider two cases: the 3-key and 2-key triple encryptions defined by

$$\begin{aligned} \text{Enc}_{K_1, K_2, K_3} &= C_{K_1} \circ C_{K_2}^{-1} \circ C_{K_3} \\ \text{Enc}_{K_1, K_2} &= C_{K_1} \circ C_{K_2}^{-1} \circ C_{K_1} \end{aligned}$$

We denote by ℓ_k the length of the K_i subkeys and by ℓ_m the block length.

3.1 3-Key Triple Encryption Case

We use the relation $\varphi(K_1, K_2, K_3) = (K_2, K_1, K_3)$. We observe that for $K = (K_1, K_2, K_3)$, we have

$$\text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1} = \left(C_{K_1} \circ C_{K_2}^{-1} \right)^2$$

The idea of the attack consists of looking for a plaintext x such that $\text{Enc}_K(x) = \text{Enc}_{\varphi(K)}(x)$. By enumerating the codebook we can find one such x with complexity 2^{ℓ_m} . Indeed, this would be a fixed point for the above permutation.

Given a random permutation over a domain of size 2^{ℓ_m} , the expected number of fixed points is 1. Additionally, there are $\frac{1}{2}$ cycles of length 2, on average. So, $C_{K_1} \circ C_{K_2}^{-1}$ has a number a of fixed points such that $E(a) = 1$ and a number b of length-2 cycles such that $E(b) = \frac{1}{2}$. In $\left(C_{K_1} \circ C_{K_2}^{-1} \right)^2$, the a fixed points are still fixed points, but elements of length-2 cycles become fixed points as well. We call them *bad fixed points*. After enumerating the codebook, we have $a + 2b$ fixed points. When $a = 0$, there are no good fixed points and it is a bad luck. This happens with probability e^{-1} . Our attack succeeds when $a > 0$, so with a success probability of $p = 1 - e^{-1}$. Note that if $a + 2b$ is odd, we are ensured that there is a good fixed point.

Assuming that x is a good fixed point, it is a fixed point of $C_{K_1} \circ C_{K_2}^{-1}$. We can enumerate all ℓ_k -bit keys and find pairs (K_1, K_2) such that $C_{K_1}(x) = C_{K_2}(x)$ with complexity 2^{ℓ_k} using a meet-in-the-middle algorithm. The correct (K_1, K_2) pair is always suggested if x is a good fixed point. Other pairs are called *wrong pairs*. We eliminate wrong pairs by using several iterations of this method.

Concretely, our attack starts as shown on Fig. 1. So, we use $r = 2n$ related keys, $d = n2^{\ell_m+1}$ broadcast

Table 2: Attacks on AES

target	parameters				complexity $C_{\text{conservative}}$	significance		reference
	data	memory	time	#keys		conservative	RK-limited	
AES-128	1	1	2^{128}	1	2^{128}			exhaustive search
AES-192	1	1	2^{192}	1	2^{192}			exhaustive search
	4	4	2^{190}	4	2^{190}			(Biham, 1996)
	2^{96}	2^{96}	2^{96}	2^{96}	2^{96}			(Biham, 1996)
	2^{123}	2^{152}	2^{176}	4	2^{176}	no	yes	(Biryukov and Khovratovich, 2009)
AES-256	1	1	2^{256}	1	2^{256}			exhaustive search
	4	4	2^{254}	4	2^{254}			(Biham, 1996)
	2^{35}	2^{35}	2^{221}	2^{35}	2^{221}			(Biham, 1996)
	2^{128}	2^{128}	2^{128}	2^{128}	2^{128}			(Biham, 1996)
	2^{131}	2^{65}	2^{131}	2^{35}	2^{131}	no	yes	(Biryukov, Khovratovich, Nikolić, 2009)
	2^{100}	2^{77}	2^{100}	4	2^{100}	yes	yes	(Biryukov and Khovratovich, 2009)

- 1: select $c_1 = 0$ and c_2, \dots, c_n at random
- 2: **for** $i = 1$ to n **do**
- 3: set a list L_i to the empty list
- 4: **repeat**
- 5: get a new BKP x with keys $K \oplus c_i$ and $\varphi(K \oplus c_i)$
- 6: let y (resp. z) be the encryption of x under key $K \oplus c_i$ (resp. $\varphi(K \oplus c_i)$)
- 7: **if** $y = z$ **then**
- 8: add y in list L_i
- 9: **end if**
- 10: **until** until all x cover the entire code book
- 11: **end for**
- 12: set I to the set of all i such that $\#L_i > 0$

Figure 1: Attack on Triple Encryption (First Part — Broadcast Known Plaintext)

- 1: select $c_1 = 0$ and c_2, \dots, c_n at random
- 2: **for** $i = 1$ to n **do**
- 3: set a list L_i to the empty list
- 4: dump the entire code book for key $K \oplus c_i$ ($\text{Enc}_{K \oplus c_i}(x)$ stored at address $h(x)$)
- 5: **repeat**
- 6: get a new KP x with key $\varphi(K \oplus c_i)$
- 7: let z be the encryption of x under key $\varphi(K \oplus c_i)$
- 8: let y to the content of cell $h(x)$
- 9: **if** $y = z$ **then**
- 10: add y in list L_i
- 11: **end if**
- 12: **until** until all x cover the entire code book
- 13: **end for**
- 14: set I to the set of all i such that $\#L_i > 0$

Figure 2: Attack on Triple Encryption (First Part — Known Plaintext)

known plaintexts, and negligible time and memory so far. There is a known plaintext variant on Fig. 2 which uses d known plaintexts $m = \ell_m 2^{\ell_m}$. Then, for each i we have a list L_i of fixed points for $(C_{K_1(i)} \circ C_{K_2(i)}^{-1})^2$. If L_i has an odd number of terms, we are ensured that there is at least one fixed point for $C_{K_1(i)} \circ C_{K_2(i)}^{-1}$ in it. Then, the attack continues as shown on Fig. 3.

The loop on $K_1(i)$ takes $t = \frac{1}{3} 2^{\ell_k}$ triple encryptions and $m = (\ell_m + \ell_k) 2^{\ell_k}$ bits of memory. The loop on $K_2(i)$ essentially takes $t = \frac{1}{3} 2^{\ell_k}$ triple encryptions (the inner loop on the found $K_2(i)$ is negligible).

The loop on (K_1, K_2, c) depends on the size of R . We denote R_n the expected number of remaining wrong keys in R using parameter n . Let $n^* - 1$ be the number of other lists which have an odd number of fixed points. We have $2^{2\ell_k}$ potential pairs but an equation to satisfy on $n^* \ell_m$ bits to end up in R . So, we have $R_n \approx 2^{2\ell_k - n^* \ell_m}$. We have

$$E(n^*) = 1 + (n-1) \sum_{a \text{ odd}} \frac{e^{-1}}{a!} = 1 + (n-1) \frac{1 - e^{-\frac{3}{2}}}{2}$$

which is nearly $0.6116 + 0.3884 \times n$. So, this loop takes $t = \frac{1+R_n}{3} 2^{\ell_k}$ triple encryptions for a good fixed point and $t = \frac{R_n}{3} 2^{\ell_k}$ for a bad one. In what follows we adjust n so that $R_n \approx 0$. Namely, for $n = 6 \frac{\ell_k}{\ell_m}$, we have $R_n \approx 2^{-0.3\ell_k - 0.6\ell_m}$ so we can neglect wrong pairs.

The main loop and the x loop iterate until it takes a good fixed point. For each L_i we have exactly i cycles of length u with probability $\frac{e^{-\frac{1}{u}}}{u! u^u}$. Let assume the probabilities for $u = 1$ and $u = 2$ are independent. For instance, the best case is $a > 1$ (some good fixed points) and $b = 0$ (no bad fixed points) with proba-

```

1: sort  $I$  in increasing order with first the list of  $i$ 's
   with  $\#L_i$  odd then the remaining ones
2: while  $I$  is not empty do
3:   pick the first  $i \in I$  and remove it from  $I$ 
4:   for all  $x$  in  $L_i$  do
5:     initialize a hash table  $H$  and a list  $R$ 
6:     for all  $K_1(i)$  do
7:       store  $(C_{K_1(i)}(x), K_1(i))$  in  $H$ 
8:     end for
9:     for all  $K_2(i)$  do
10:    for each  $K_1(i)$  such that
       $(C_{K_2(i)}(x), K_1(i)) \in H$  do
11:      compute  $K_1$  and  $K_2$  from  $K_1(i)$  and
       $K_2(i)$  using  $c_i$  and set  $c = 0$ 
12:      for each  $j \in I$  do
13:        compute  $K_1(j)$  and  $K_2(j)$  from  $K_1$ 
        and  $K_2$  using  $c_j$ 
14:        look if there is  $y \in L_j$  such that
         $C_{K_1(j)}(y) = C_{K_2(j)}(y)$ 
15:        if there is such  $y$  then
16:          increment  $c$ 
17:        end if
18:        if there is no such  $y$  and  $j$  is odd then
19:          exit the  $j$  loop and set  $c$  to 0
20:        end if
21:      end for
22:      if  $c \neq 0$ , add  $(K_1, K_2, c)$  in list  $R$  sorted
      by decreasing  $c$ 
23:    end for
24:  end for
25: end while
26: for each  $(K_1, K_2, c) \in R$  sorted by  $c$  do
27:   for all  $K_3$  do
28:    if  $(K_1, K_2, K_3)$  consistent with data then
29:      yield  $(K_1, K_2, K_3)$  and exit
30:    end if
31:  end for
32: end for
33: end while
34: attack failed

```

Figure 3: Attack on 3-Key Triple Encryption (Second Part)

bility $e^{-\frac{1}{2}}(1 - e^{-1}) \approx 0.38$. We denote by N_n (resp. N_n^*) the expected number of iterations of the i and x loops (resp. in the case that the attack succeeds). In the case of failure ($a = 0$), we have $2b$ iterations so the expected number is 1 for each list. That is, the expected number of iterations before the attack fails is n . Since this happens with probability e^{-n} , we have

$$N_n^* = \frac{N_n - ne^{-n}}{1 - e^{-n}}$$

Given $a > 0$ good fixed points and $2b$ bad ones, the

expected number of iterations is

$$N_n(a, b) = \frac{1}{\binom{a+2b}{a}} \sum_{i=1}^{2b+1} i \binom{a+2b-i}{a-1}$$

which does not depend on n . For $a = 0$, it is of $N_n(0, b) = 2b + N_{n-1}$ which does depend on n . Furthermore, we have

$$N_n = \sum_{a,b} N_n(a, b) \frac{e^{-\frac{3}{2}}}{a!2^b b!}$$

So,

$$\begin{aligned} N_n &= \sum_b (2b + N_{n-1}) \frac{e^{-\frac{3}{2}}}{2^b b!} + \sum_{a>0,b} N_n(a, b) \frac{e^{-\frac{3}{2}}}{a!2^b b!} \\ &= (N_{n-1} + 1)e^{-1} + \sum_{a>0,b} N_n(a, b) \frac{e^{-\frac{3}{2}}}{a!2^b b!} \end{aligned}$$

where the sum does not depend on n . We computed some values of N_n in Table 3. So, the number of iterations can be approximated to 2 in any success case. Finally, the total complexity is of

$$\begin{aligned} r &= 2n \\ d &= n2^{\ell_m+1} \\ t &= \begin{cases} 2\left(\frac{2}{3}2^{\ell_k} + \frac{1}{3}2^{\ell_k}R_n\right) + \frac{1}{3}2^{\ell_k} & \text{if success} \\ t = n\left(\frac{2}{3}2^{\ell_k} + \frac{1}{3}2^{\ell_k}R_n\right) & \text{if failure} \end{cases} \\ m &= \begin{cases} (\ell_m + \ell_k)2^{\ell_k} & \text{for CP variant} \\ \max(\ell_m 2^{\ell_m}, (\ell_m + \ell_k)2^{\ell_k}) & \text{for KP variant} \end{cases} \\ p &= 1 - e^{-n} \end{aligned}$$

In general, we suggest $n \approx 6\frac{\ell_k}{\ell_m}$ to get $t = \frac{5}{3}2^{\ell_k}$ in a success case.

In the case of DES, we have $\ell_k = 56$ and $\ell_m = 64$. We take $n = 3$ to get $n^* \approx 1.777$ and $R_n \approx 2^{-1.72}$. So, we use $r = 6$ keys. We use $d = 2^{67}$ chosen plaintexts or ciphertexts, or known plaintexts. The time complexity is $t = 2^{57}$ triple encryptions in the all cases. The memory complexity is $m = 2^{63}$ bits in the chosen message variant and $m = 2^{72}$ in the known plaintext variant. The key to recover has 168 bits. The attack succeeds with probability $p = 95\%$. Note that this attack is better than the generic related-key attack using tradeoffs. It works in the ideal cipher model. Bellare and Rogaway proved that the best (non-related-key) generic attack in the ideal cipher model would require at least 2^{78} encryptions (Bellare and Rogaway, 2006). This example shows that the result no longer holds in the related-key model.

In the case we would like to use a triple AES encryption, we obtain different results which are summarized in Table 4.

Table 3: Some values for the expected number of iterations N_n and N_n^*

n	1	2	3	4	5	6	7	8	9	10
N_n	1.264	1.729	1.900	1.963	1.987	1.995	1.998	1.999	2.000	2.000
N_n^*	1.418	1.687	1.843	1.925	1.966	1.985	1.994	1.997	1.999	2.000

Comparison with the Kelsey-Schneier-Wagner attack. Kelsey, Schneier, and Wagner presented a related-key attack against 3-key triple encryption which has similar performances (Kelsey, Schneier, Wagner, 1996). It consists in using

$$\varphi(K_1, K_2, K_3) = (K_1 \oplus \Delta, K_2, K_3)$$

Then, $\text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1} = C_{K_1} \circ C_{K_1 \oplus \Delta}^{-1}$ which only depends on K_1 . So, exhaustive search can recover K_1 . For DES, this attack has $r = 2$, $d = 2$ (known and chosen plaintexts), $t = 2^{56}$ encryptions, $m = 2^{56}$ bits, and $p = 100\%$. So, it is better than our attack. Contrarily to ours, it has no extension to 2-key triple encryption. However, this attack extends to the encrypt-encrypt-encrypt triple encryption mode whereas our attack restricts to the encrypt-decrypt-encrypt construction.

Note that getting $\text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1}(y) = z$ on a random y is equivalent to getting $\text{Enc}_K(x) = y$ and $\text{Enc}_{\varphi(K)}(x) = z$ on a random x . So, this attack is in the BKP model.

Comparison with the Phan attack. In the category of known plaintext attacks, Phan (Phan, 2004) uses

$$\varphi(K_1, K_2, K_3) = (K_1, K_3, K_2)$$

(which is similar to our relation) and a slide attack. It breaks 3-key triple encryption using $r = 2$, $d = 2^{33}$ (known and chosen plaintexts), $t = \frac{1}{3}2^{88}$ triple encryptions, and $m = 2^{38}$ bits. This attack extends to encrypt-encrypt-encrypt and to 2-key triple encryption (with a memory complexity inflated to $m = 2^{94.5}$ bits). Our known plaintext attack uses a quite lower time complexity but a higher number of chosen plaintexts.

3.2 2-Key Triple Encryption Case

We use the relation $\varphi(K_1, K_2) = (K_2, K_1)$. We observe that for $K = (K_1, K_2)$, we have

$$\text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1} = (C_{K_1} \circ C_{K_2}^{-1})^3$$

Fixed points of $\text{Enc}_K \circ \text{Enc}_{\varphi(K)}^{-1}$ are fixed points of $C_{K_1}(x) \circ C_{K_2}^{-1}$ or points in cycles of length 3. We just proceed as in the previous attack. The probability to have a fixed points and b cycles of length 3 is $\frac{e^{-\frac{4}{3}}}{a!3^b b!}$.

So, $E(a) = 1$ and $E(b) = \frac{1}{3}$. The number of values x obtained is $a + 3b$. If it is no multiple of 3 then we have a good fixed point for sure.

The final complexity is very similar to the 3-key encryption case. The difference is that we no longer need the exhaustive search on K_3 and wrong (K_1, K_2) pairs are discarded by a simple consistency check. We can work with $n = 1$ and $p = 63\%$. The formulas become

$$\begin{aligned} N_n(a, b) &= \frac{1}{\binom{a+3b}{a}} \sum_{i=1}^{3b+1} i \binom{a+3b-i}{a-1} \\ N_n(0, b) &= 3b + N_{n-1} \\ N_n &= \sum_{a,b} N_n(a, b) \frac{e^{-\frac{4}{3}}}{a!3^b b!} \\ &= (N_{n-1} + 1)e^{-1} + \sum_{a>0, b} N_n(a, b) \frac{e^{-\frac{4}{3}}}{a!3^b b!} \\ N_n^* &= \frac{N_n - ne^{-n}}{1 - e^{-n}} \end{aligned}$$

The new table for N_n gives identical results. So, the new complexity is

$$\begin{aligned} r &= 2n \\ d &= n2^{\ell_m+1} \\ t &= \begin{cases} \frac{4}{3}2^{\ell_k} & \text{if success} \\ t = \frac{2n}{3}2^{\ell_k} & \text{if failure} \end{cases} \\ m &= \begin{cases} (\ell_m + \ell_k)2^{\ell_k} & \text{for CP variant} \\ \max(\ell_m 2^{\ell_m}, (\ell_m + \ell_k)2^{\ell_k}) & \text{for KP variant} \end{cases} \\ p &= 1 - e^{-n} \end{aligned}$$

The first part of the algorithm works like in the 3-key case with two variants on Fig. 1 and Fig. 2. The second part of the algorithm is shown on Fig. 4.

In the case of DES, we take $n = 1$, so $r = 2$. We use $d = 2^{65}$ chosen plaintexts or ciphertexts. The time complexity is $t = 2^{56}$ triple encryptions in the all cases. The memory complexity is $m = 2^{63}$ bits and the key to recover has 112 bits. The known plaintext variant uses $d = 2^{65}$ known plaintexts and the memory complexity becomes $m = 2^{72}$ bits. The attack succeeds with probability $p = 63\%$. Comparison with other attacks is presented in Table 1.

Table 4: Semi-Generic Attack against Triple Encryption (Chosen Message Variant)

cipher	2-key				3-key			
	DES	AES128	AES192	AES256	DES	AES128	AES192	AES256
key size	116	256	384	512	168	384	576	768
ℓ_k	56	128	192	256	56	128	192	256
ℓ_m	64	128	128	128	64	128	128	128
#keys	2	2	2	2	6	8	14	18
#chosen plaintexts	2^{65}	2^{129}	2^{129}	2^{129}	2^{67}	2^{131}	2^{132}	2^{132}
time complexity	2^{56}	2^{128}	2^{192}	2^{256}	2^{57}	2^{129}	2^{193}	2^{257}
memory complexity	2^{63}	2^{136}	2^{200}	2^{255}	2^{63}	2^{136}	2^{200}	2^{265}
success probability	63%	63%	63%	63%	95%	98%	100%	100%
$C_{\text{conservative}}$	2^{66}	2^{136}	2^{200}	2^{265}	2^{67}	2^{136}	2^{200}	2^{265}

```

1: sort  $I$  in increasing order with first the list of  $i$ 's
   with  $\#L_i$  not multiple of 3 then the remaining ones
2: while  $I$  is not empty do
3:   pick the first  $i \in I$  and remove it from  $I$ 
4:   for all  $x$  in  $L_i$  do
5:     initialize a hash table  $H$ 
6:     for all  $K_1(i)$  do
7:       store  $(C_{K_1(i)}(x), K_1(i))$  in  $H$ 
8:     end for
9:     for all  $K_2(i)$  do
10:      for each  $K_1(i)$  such that
         $(C_{K_2(i)}(x), K_1(i)) \in H$  do
11:        compute  $K_1$  and  $K_2$  from  $K_1(i)$  and
           $K_2(i)$  using  $c_i$  and set  $c = 0$ 
12:        if  $(K_1, K_2, K_3)$  consistent with data
          then
13:          yield  $(K_1, K_2, K_3)$  and exit
14:        end if
15:      end for
16:    end for
17:  end for
18: end while
19: attack failed

```

Figure 4: Attack on 2-Key Triple Encryption (Second Part)

Comparison with the Merkle-Hellman meet-in-the-middle attack. Merkle and Hellman proposed to use a simple collision algorithm to find collisions between the list of all $C_{K_2}^{-1}(0)$ and the list of all $C_{K_1}^{-1}(\text{Enc}(C_{K_1}^{-1}(0)))$ (Merkle and Hellman, 1981). This requires to encrypt all chosen plaintexts $C_{K_1}^{-1}(0)$. A variant with known plaintexts only can be done as follows: first we make a dictionary of all $(C_{K_1}(0), K_1)$ in addition to the dictionary of all $(C_{K_2}^{-1}(0), K_2)$. Then, every time we receive a plaintext/ciphertext pair (x, y) we look if x is in the first dictionary to find K_1 . If it is, we compute $z = C_{K_1}^{-1}(y)$ and get a new element $C_{K_1}^{-1}(\text{Enc}(C_{K_1}^{-1}(0))) = z$. We can look for z in the sec-

ond dictionary. This doubles the memory complexity and increase the data complexity to essentially the entire code book.

This attack has lower complexity parameters than ours for the chosen plaintext variant. The known plaintext variants are equivalent (except that we use related keys and the Merkle-Hellman attack does not).

4 CONCLUSION

We presented a new attack on triple-encryption which uses related keys. It can use chosen messages or known plaintexts but require the entire code book for related keys. Our attack is the best in the known plaintext attack category in the 3-key case. Besides, the best attacks remain the Merkle-Hellman attack (Merkle and Hellman, 1981) in the 2-key case and the Kelsey-Schneier-Wagner attack (Kelsey, Schneier, Wagner, 1996) in the 3-key case.

REFERENCES

- M. Bellare, P. Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology EUROCRYPT'06*, St. Petersburg, Russia, Lecture Notes in Computer Science 4004, pp. 409–426, Springer-Verlag, 2006.
- E. Biham. New Types of Cryptanalytic Attacks Using related Keys. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norway, Lecture Notes in Computer Science 765, pp. 398–409, Springer-Verlag, 1994.
- E. Biham. New Types of Cryptanalytic Attacks Using Related Keys. *Journal of Cryptology*, vol. 7, pp. 229–246, 1994.

- E. Biham. How to Decrypt or even Substitute DES-Encrypted Messages in 2^{28} steps. Technical report CS 884, 1996.
- A. Biryukov, D. Khovratovich. Related-key Cryptanalysis of the Full AES-192 and AES-256. In *Advances in Cryptology ASIACRYPT'09*, Tokyo, Japan, Lecture Notes in Computer Science 5912, pp. 1–18, Springer-Verlag, 2009.
- A. Biryukov, D. Khovratovich, I. Nikolić. Distinguisher and Related-Key Attack on the Full AES-256. In *Advances in Cryptology CRYPTO'09*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 5677, pp. 231–249, Springer-Verlag, 2009.
- W. Diffie, M.E. Hellman. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, vol. 10, pp. 74–84, 1977.
- J. Kelsey, B. Schneier, D. Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In *Advances in Cryptology CRYPTO'96*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 1109, pp. 237–251, Springer-Verlag, 1996.
- L.R. Knudsen. Cryptanalysis of LOKI91. In *Advances in Cryptology AUSCRYPT'92*, Gold Coast, Queensland, Australia, Lecture Notes in Computer Science 718, pp. 196–208, Springer-Verlag, 1993.
- S. Lucks. Attacking Triple Encryption. In *Fast Software Encryption'98*, Paris, France, Lecture Notes in Computer Science 1372, pp. 239–253, Springer-Verlag, 1998.
- R.C. Merkle, M.E. Hellman. On the Security of Multiple Encryption. *Communications of the ACM*, vol. 24, pp. 465–467, 1981.
- P.C. van Oorschot, M.J. Wiener. A Known-Plaintext Attack on Two-Key Triple Encryption. In *Advances in Cryptology EUROCRYPT'90*, Aarhus, Denmark, Lecture Notes in Computer Science 473, pp. 318–325, Springer-Verlag, 1991.
- P.C. van Oorschot, M.J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *Journal of Cryptology*, vol. 12, pp. 1–28, 1999.
- R. C.-W. Phan. Related-Key Attacks on Triple-DES and DESX Variants. In *Topics in Cryptology CT-RSA'04*, San Francisco, California, U.S.A., Lecture Notes in Computer Science 2964, pp. 15–24, Springer-Verlag, 2004.
- M.J. Wiener. The Full Cost of Cryptanalytic Attacks. *Journal of Cryptology*, vol. 17, pp. 105–124, 2004.