A Related-Key Attack against Multiple Encryption based on Fixed Points[⋆]

Aslı Bay, Atefeh Mashatan[⋆⋆], and Serge Vaudenay

EPFL, Switzerland
email{asli.bay, atefeh.mashatan , serge.vaudenay}@epfl.ch

Abstract. In order to alleviate the burden of short keys, encrypting a multiple times has been proposed. In the multiple encryption mode, there may be encryptions under the same or different keys. There have been several attacks against this encryption mode. When triple encryption is based on two keys, for instance, Merkle and Hellman proposed a subtle meet-in-the-middle attack with a complexity similar to breaking a single encryption, requiring nearly all the codebook. In the case of triple encryption with three keys, Kelsey, Schneier, and Wagner proposed a related-key attack with complexity similar to breaking a single encryption.
In this paper, we propose a new related-key attack against triple encryption which compares to breaking single encryption in the two aforementioned cases. Based on finding fixed points in a decrypt-encrypt sequence, we propose a related-key attack against a two-key triple encryption. Our attack has exactly the same performance as a meet-in-the-middle on double encryption. When considering two keys, it is comparable to the Merkle-Hellman attack, except that uses related keys. And, when considering three keys, it has a higher complexity than the Kelsey-Schneier-Wagner attack, but has the advantage that it 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 inevitably broken by generic attacks such as exhaustive search, we use these attacks as reference 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 [4, 3, 10] proposed the notion of related-key attacks in which an adversary can impose to change the secret key following some chosen relation φ. Related-key attacks open a way to new generic attacks

[⋆] This is a full version of [15].

[⋆⋆] This author was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center of the SNF under grant number 5005-67322.

such as the ones by Biham [5]. Therefore, exhaustive search may no longer be the reference for assessing the security of a cipher.

As an example of a related-key attack, Kelsey, Schneier, and Wagner [9] presented an attack against three-key triple encryption which shows that it is not more secure than single encryption.

Notations. In this paper, KP, BKP, CP, and CC denote known plaintexts, broadcast known plaintexts, chosen plaintexts, and chosen ciphertexts, respectively. In the BKP model, the adversary obtains a random plaintext and its encryption under different keys. In addition, RK denotes the related-key model where the adversary either knows or chooses the relation between the unknown keys. On the other hand, $d_{\mathsf{KP}}, d_{\mathsf{BKP}}$, and $d_{\mathsf{CP}}, d_{\mathsf{CC}}$ denote respective data complexities of KP, BKP, CP, and CC attacks and C_K and C_K^{-1} denote block cipher encryption and decryption under K, respectively. Furthermore, $E(X)$ denotes expected value of a random variable X and $\#L$ denotes the cardinality of a set L, and $\oplus$ denotes exclusive or.

Related work on Triple-DES. As far as we know, the only related-key attacks against Triple-DES are the generic attack of Biham, the Kelsey-Schneier-Wagner attack, and an attack of Phan [5, 9, 12]. 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 [1, 11, 13, 14]. We use the table of Phan [12] given in Table 1 to compare these attacks with ours. Note that the results are given in sightly different units, that is, 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 aforementioned attacks will be discussed in this paper.

Our contribution. In this paper, we first formalize the various ways to compare the complexity of related-key attacks. Following a full-cost model, an attack is significant if $tm/p < 2^{\ell}$, 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 $\max(t/p, m)$ with $2^{\ell/2}$. We can also consider comparison in a restricted attack model in order to limit some characteristics such as the number of related keys.

We then present a new attack on triple encryption which is based on the discovery of fixed points for the mapping

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

for some relation φ. This discovery requires the entire codebook in a *Broadcast Known Plaintext (BKP) attack* for Enc_K and $\mathsf{Enc}_{\varphi(K)}$ which makes our data complexity high. Once we have a (good) fixed point, our attack becomes similar to a standard meet-in-the-middle attack. Hence, it has a pretty low complexity. Finally, we show that our attack has a comparable complexity to the best ones so far. In the two-key case, it becomes the best known-plaintext attack. In the sequel, we discuss the comparison of related-key attacks in different models.

Table 1. Summary of attacks against Triple-DES.

	parameters				complexity	
target	data	memory	time	#keys	$C_{\mathsf{conservative}}$	reference
Two-Key Triple-DES	2 (KP)	2^{7}	2^{112}	1	2^{112}	exhaustive search
	2^{56} (CP)	2^{62}	2^{56}	2^{56}	2^{62}	[5] (generic)
	2^{56} (KP)	2^{58}	2^{112}	1	2^{112}	[13, 14]
	2^{33} (KP)	$2^{91.5}$	2^{86}	2	$2^{91.5}$	[12]
	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}	[1] variant
	2^{56} (CP)	2^{63}	2^{56}	1	2^{63}	[1]
Three-Key Triple-DES	3 (KP)	2^{8}	2^{168}	1	2^{168}	exhaustive search
	2^{84} (CP)	2^{92}	2^{84}	2^{84}	2^{92}	[5] (generic)
	2^{32} (KP)	2^{90}	2^{104}	1	2^{104}	[11]
	3 (CP)	2^{58}	2^{110}	1	2^{110}	[1]
	2^{33} (KP)	2^{35}	2^{86}	2	2^{86}	[12]
	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}	[9]

2 Comparing Related-Key Attacks

Given a dedicated attack against a cipher, it is common to compare it with exhaustive search and declare the cipher broken if the attack is more efficient. However, it is unfair to do this, since the attack model may already have better generic attacks than exhaustive search. More expilicitly, we should consider other generic attacks in the attack model while comparing the efficiency of the given attack.

As an example, Biham's generic attack [5] applies standard time-memory tradeoffs in the related-key model. His attack consists of collecting $y_i = \mathsf{Enc}_{K_i}(x)$ for a fixed plaintext x and r related keys K_i, $1 \leq i \leq r$. Hence, we use r chosen plaintexts. Then, it builds a dictionary (y_i, i) and runs a multi-target key recovery to find one key K such that $\mathsf{Enc}_K(x)$ 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

– the number of related keys r (the number of keys which are involved is r, i.e., $r = 1$ if 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 we may further distinguish quick-access, or slow-access memory, read/write memory or read-only memory; and
– the probability of success p.

There are many other possible refinements. We can compare attacks by using the partial ordering p on vectors $(r, d, t, m, 1/p)$, i.e.,

$$(r, d, t, m, p) \leq_{\mathsf{p}} (r', d', t', m', p') \\ \Updownarrow \\ r \leq r' \text{ and } d \leq d' \text{ and } t \leq t' \text{ and } m \leq m' \text{ and } p \geq p'.$$

When a category such as the data complexity d has a sub-characterization $(d_{\mathsf{KP}}, d_{\mathsf{BKP}}, d_{\mathsf{CP}}, d_{\mathsf{CC}})$, then $d \leq d'$ implies another 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. Since it is not always possible to compare two multi-dimensional vectors, whether an attack is significant or not is not always clear. Therefore, it is quite common to extend the partial ordering $\leq_{\mathsf{p}}$ using different models which are discussed below.

Conservative model. Traditionally, t, m, and p are combined into a "complexity" which is arbitrarily measured by $\max(t/p, m)$. We could equivalently adopt $t/p + m$ 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 $1/p$ sessions of the attack. So, t has a factor $1/p$ corresponding to $1/p$ 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

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

For instance, $2^{\ell/2}$ is the complexity of the Biham attacks [5].[1]

In some cases, there may be a special treatment for r and d though, especially regarding the $1/p$ factor. Actually, the current $1/p$ factor corresponds to

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

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 r/p. Therefore, we can keep this in mind that the $C_{\mathsf{conservative}}$ 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 rather convenient because two attacks can always be compared by $C_{\mathsf{conservative}}$: let

$$(r,d,t,m,p) \leq_{\mathsf{conservative}} (r',d',t',m',p')$$
$$\Updownarrow$$
$$C_{\mathsf{conservative}}(r,d,t,m,p) \leq C_{\mathsf{conservative}}(r',d',t',m',p').$$

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

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

Limited related-key models. Arguably, related keys (or even chosen plaintexts or ciphertexts) are harder to obtain, compared to the time spent in the attack. More explicitly, getting encryption or decryption of the data under different related keys that are chosen by the adversary requires much more work than doing computations in the attack. For instance, an attack with complexity $2^{3\ell/4}$, $r=1$, and $d=1$ is declared not significant with $\leq_{\mathsf{conservative}}$ because of the Biham attack [5] with complexity $2^{\ell/2}$, which is a bit unfair. Therefore, we could either go back to some partial ordering or to some attack model restrictions. For example, 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 not more related keys than the attack has. 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 less number of related keys than the attack has. If (r,d,t,m,p) is the complexity vector of the attack, we shall compare $C_{\mathsf{conservative}}(r,d,t,m,p)$ with the one of $(r',r',t',r',1-e^{-r't'2^{-\ell}})$ 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

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

Other limited models. We may also consider other limited models. For instance, we can restrict ourselves 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 [16] introduced the full cost expressed as $\mathcal{O}(t + tm/c + t\sqrt{c\rho^3})$, where c is the number of processors and ρ is the rate of access to the memory of all processors per time unit. (We assume here that parameters are normalized so that we can assume $p = 1$.) Using a single processor and $\rho = 1$, this simplifies to $\mathcal{O}(tm)$. Again, we replace t by $\max(r, d, t)$ to integrate r and d. Therefore, we define the full cost as

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

and define

$$\begin{gathered}(r, d, t, m, p) \leq_{\mathsf{full}} (r', d', t', m', p') \\ \Updownarrow \\ C_{\mathsf{full}}(r, d, t, m, p) \leq C_{\mathsf{full}}(r', d', t', m', p').\end{gathered}$$

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

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

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

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 [8] with a key of size ℓ_k has a full cost of $2^{4\ell_k/3}$ and may also be reduced to $2^{6\ell_k/5}$ using parallelism [16]. 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 [6] 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 [7] 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 [6] on AES-256 is significant for both criteria.

3 Semi-Generic Related-Key Attacks against Triple-DES

We propose here a related-key attack against Triple-DES. For the three-key triple encryption case, this attack 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 Triple-DES, that is, the encrypt-decrypt-encrypt structure. For the two-key

Table 2. Attacks on AES

	parameters				complexity	significance		
target	data	memory	time	#keys	$C_{\mathsf{conservative}}$	conservative	RK-limited	reference
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}			[5]
	2^{96}	2^{96}	2^{96}	2^{96}	2^{96}			[5]
	2^{123}	2^{152}	2^{176}	4	2^{176}	no	yes	[6]
AES-256	1	1	2^{256}	1	2^{256}			exhaustive search
	4	4	2^{254}	4	2^{254}			[5]
	2^{35}	2^{35}	2^{221}	2^{35}	2^{221}			[5]
	2^{128}	2^{128}	2^{128}	2^{128}	2^{128}			[5]
	2^{131}	2^{65}	2^{131}	2^{35}	2^{131}	no	yes	[7]
	2^{100}	2^{77}	2^{100}	4	2^{100}	yes	yes	[6]

case, it is generic since it also works for the encrypt-encrypt-encrypt structure. The three-key and the two-key triple encryptions defined by

$$\mathsf{Enc}_{K_1,K_2,K_3} = C_{K_1} \circ C^{-1}_{K_2} \circ C_{K_3}, \text{and}$$
$$\mathsf{Enc}_{K_1,K_2} = C_{K_1} \circ C^{-1}_{K_2} \circ C_{K_1}.$$

We denote by ℓ_k the length of the K_i subkeys and by ℓ_m the block length. We also consider the two-key triple encryption with the encrypt-encrypt-encrypt structure, i.e., $\mathsf{Enc}'_{K_1,K_2} = C_{K_1} \circ C_{K_2} \circ C_{K_1}$.

3.1 Preliminaries

We will give some necessary background regarding random permutations without providing their proofs due to the space limitation.

Definition 1. *Let π be a permutation over a finite set S. The k-cycles of π is $(c_1, c_2, \ldots, c_k)$ such that $\pi(c_i) = c_{i+1}$ for $i = 1, \ldots, k-1$ and $\pi(c_k) = c_1$.*

For example, when $k = 1$, it is a 1-cycle, i.e., $\pi(c) = c$ and c is also called a *fixed point.* When $k = 2$, it is 2-cycles such that $\pi(c_1) = c_2$ and $\pi(c_2) = c_1$.

Lemma 2. *Let π be a permutation over a finite set S. The probability of having* exactly *t k-cycles is $e^{-1/k}/k^t t!$ when the cardinality of S grows to infinity.*

For instance, the probability of having no fixed points is e^{-1} which is computed by substituting $t = 0$ and $k = 1$. On the other hand, given a random permutation, the expected number of fixed points (1-cycles) is 1. Additionally, the expected number of 2-cycles is 1/2. This is generalized in the following lemma.

Lemma 3. *Let π be a permutation over a finite set S. The expected number k-cycles tends towards $1/k$ as the cardinality of S grows to infinity.*

3.2 Three-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

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

The idea of the attack consists of looking for a plaintext x such that $\mathsf{Enc}_K(x) = \mathsf{Enc}_{\varphi(K)}(x)$. 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.

Under heuristic assumptions, the permutation $C_{K_1} \circ C^{-1}_{K_2}$ has a number of fixed points such that $E(a) = 1$ and b number of 2-cycles such that $E(b) = 1/2$. Since, $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^2$ is a composition of $C_{K_1} \circ C^{-1}_{K_2}$ with itself, the a fixed points of $C_{K_1} \circ C^{-1}_{K_2}$ are the fixed points of $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^2$, too. However, the elements of 2-cycles of $C_{K_1} \circ C^{-1}_{K_2}$ become fixed points for $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^2$, as well. So, we have $a + 2b$ fixed points. In the attack, we take advantage of fixed points of $C_{K_1} \circ C^{-1}_{K_2}$. Hence, we call the fixed points of $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^2$ which is also the fixed points of $C_{K_1} \circ C^{-1}_{K_2}$ as *good fixed points* and the elements of 2-cycles of $C_{K_1} \circ C^{-1}_{K_2}$ as *bad fixed points.*

```
 1: select $c_1 = 0$ and $c_2, \ldots, c_n$ 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 all $x$ cover the entire codebook
11: end for
12: set $I$ to the set of all $i$ such that $\#L_i > 0$
```

Fig. 1. Attack on Triple Encryption (First Part — Broadcast Known Plaintext)

The attack is composed of two parts, namely, the fixed points finding part and the key recovery part. Our attack starts as shown in Fig. 1 or in Fig. 2 depending on the type of the data set (BKP or KP). In the BKP variant, we first determine the relation between related key pairs as $c_1 = 0$ and $c_2, \ldots, c_n$, at random. Then, for each i we have a list L_i of fixed points for $\left(C_{K_1(i)} \circ C^{-1}_{K_2(i)}\right)^2$ by enumerating the codebook. If the cardinality $\#L_i$ of L_i is nonzero, then we keep the index i of L_i in I. Note that, if L_i has an odd number of terms, we ensure that there is at least one fixed point for $C_{K_1(i)} \circ C^{-1}_{K_2(i)}$ in it.

```
select $c_1 = 0$ and $c_2, \ldots, c_n$ at random
for $i = 1$ to $n$ do
    set a list $L_i$ to the empty list
    dump the entire codebook for key $K \oplus c_i$ ($\mathsf{Enc}_{K \oplus c_i}(x)$ stored at address $h(x)$)
    repeat
        get a new KP $x$ with key $\varphi(K \oplus c_i)$
        let $z$ be the encryption of $x$ under key $\varphi(K \oplus c_i)$
        let $y$ to the content of cell $h(x)$
        if $y = z$ then
            add $y$ in list $L_i$
        end if
    until all $x$ cover the entire codebook
end for
set $I$ to the set of all $i$ such that $\#L_i > 0$
```

Fig. 2. Attack on Triple Encryption (First Part — Known Plaintext)

Then, the attack continues as shown in Fig. 3. Starting from the first i in I, we pick every fixed point x from the list L_i and 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. Then, for each remaining j in I, we keep counter c in order to determine whether (K_1, K_2) suggested by the meet-in-the-middle algorithm is right key pair or not. Notice that the correct (K_1, K_2) pair is always suggested if x is a good fixed point. Other pairs are called *wrong pairs*. In order to eliminate wrong pairs, we use the list of other fixed points by checking whether $C_{K_1(j)}(y) = C_{K_2(j)}(y)$ or not, where $K_1(j) = K_1 \oplus c_j$ and $K_2(j) = K_2 \oplus c_j$. If there exists such y, then we increment c by 1. Otherwise, if there is no such y and $\#L_i$ is odd, then we decide that this x is not a good fixed point. In addition, we make another list R to keep promising key pairs which have nonzero counter, namely, (K_1, K_2, c) and $c \neq 0$. Finally, we make exhaustive search on K_3 with the promising key pairs (K_1, K_2) existing in the list R. If there is no true K_3, we can make several iterations of this method until going through all lists with non-zero cardinality.

Success Probability. Our attack succeeds when there is one related key pair having at least one good fixed point, i.e., $a > 0$. From Lemma 2, for a random permutation, the probability of having no fixed point is e^{-1}. Therefore, the attack fails when there is no good fixed point for each related key pair which happens with probability e^{-n}. Hence, the success probability of the attack is $p = 1 - e^{-n}$.

Complexity Analysis. In order to generate fixed points in Fig. 1, we use $r = 2n$ related keys, $d = n2^{\ell_m+1}$ broadcast known plaintexts, and negligible time and memory. For the known plaintext variant depicted in Fig. 2, data complexity is the same as the BKP variant, but the memory complexity is higher, i.e., $m = \ell_m 2^{\ell_m}$.

```
sort $I$ in increasing order with first the list of $i$'s with $\#L_i$ odd then the remaining ones
while $I$ is not empty do
  pick the first $i \in I$ and remove it from $I$
  for all $x$ in $L_i$ do
    initialize a hash table $H$ and a list $R$
    for all $K_1(i)$ do
      store $(C_{K_1(i)}(x), K_1(i))$ in $H$
    end for
    for all $K_2(i)$ do
      for each $K_1(i)$ such that $(C_{K_2(i)}(x), K_1(i)) \in H$ do
        compute $K_1$ and $K_2$ from $K_1(i)$ and $K_2(i)$ using $c_i$ and set $c = 0$
        for each $j \in I$ do
          compute $K_1(j)$ and $K_2(j)$ from $K_1$ and $K_2$ using $c_j$
          look if there is $y \in L_j$ such that $C_{K_1(j)}(y) = C_{K_2(j)}(y)$
          if there is such $y$ then
            increment $c$
          end if
          if there is no such $y$ and $\#L_j$ is odd then
            exit the $j$ loop and set $c$ to 0
          end if
        end for
        if $c \neq 0$, add $(K_1, K_2, c)$ in list $R$ sorted by decreasing $c$
      end for
    end for
  end for
  for each $(K_1, K_2, c) \in R$ sorted by $c$ do
    for all $K_3$ do
      if $(K_1, K_2, K_3)$ consistent with data then
        yield $(K_1, K_2, K_3)$ and exit
      end if
    end for
  end for
end while
attack failed
```

Fig. 3. Attack on Three-Key Triple Encryption (Second Part)

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

The loop in (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

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

where $\sum_{a \text{ odd}} 1/a! = (e - e^{-1})/2$. Then, we have $E(n^*) \approx 0.4323 \times n + 0.5677$. So, this loop takes $t = (1 + R_n)2^{\ell_k}/3$ triple encryptions for a good fixed point and $t = R_n 2^{\ell_k}/3$ for a bad one. In what follows, we adjust n so that $R_n \approx 0$. Namely, for $n = 6\ell_k/\ell_m$, we have $R_n \approx 2^{-0.4\ell_k - 0.6\ell_m}$ 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 t k-cycles with probability $e^{-1/k}/k^t t!$. Let us assume that the probabilities for $k = 1$ and $k = 2$ are independent. For instance, the best case is $a > 1$ (some good fixed points) and $b = 0$ (no bad fixed points) with probability $e^{-1/2}(1-e^{-1}) \approx 0.38$. Let N_n (resp. N_n^*) denote 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, hence, the expected number of bad fixed point is 1 for each list That is, since there are n lists, the expected number of iterations before the attack fails is n. This happens with probability e^{-n}, therefore, we have

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

Let (a_i, b_i) be the numbers of 1-cycles and 2-cycles in the lists in I, respectively, where i is at most n. Regarding the first list, for a_1 nonzero good fixed points and $2b_1$ bad ones, the expected number of iterations is

$$N_n(a_1, b_1) = \frac{1}{\binom{a_1+2b_1}{a_1}} \sum_{i=1}^{2b_1+1} i \binom{a_1 + 2b_1 - i}{a_1 - 1},$$

which does not depend on n.

Notice that for $a_1 = 0$, it is $N_n(0, b_1) = 2b_1 + \bar{N}_{n-1}$, where $\bar{N}_{n-1}$ is the expected number of iterations conditioned to that all a_i's are even. This is because, we first consider the lists having odd number of a_i's in the algorithm. Therefore, if $a_1 = 0$, then the remaining lists will contain even number of a_i's. Hence, we will continue searching points in the remaining lists with all bad points of the first list. Furthermore, we have

$$N_n = \sum_{a_1,\ b_1} N_n(a_1, b_1)\,\Pr[a_1, b_1].$$

Here, we compute the joint probability $\Pr[a_1, b_1]$ as

$$\Pr[a_1, b_1] = \Pr[a_1, b_1 | a_1 \text{ odd}] \Pr[a_1 \text{ odd}] + \Pr[a_1, b_1 | a_1 \text{ even}] \Pr[a_1 \text{ even}].$$

Then, the probability that a_1 is even is computed as

$$\Pr[a_1 \text{ even}] = \Pr[a_1, \ldots, a_n \text{ even}] = \left(\sum_i \frac{e^{-1}}{(2i)!} \right)^n = \left(\frac{1 + e^{-2}}{2} \right)^n.$$

We can equivalently write the probability that a_1 is odd as

$$\Pr[a_1 \text{ odd}] = 1 - \Pr[a_1 \text{ even}] = 1 - \left(\frac{1 + e^{-2}}{2} \right)^n.$$

On the other hand, for a_1 odd, we have

$$\Pr[a_1, b_1 | a_1 \text{ odd}] = \frac{\frac{e^{-\frac{3}{2}}}{a_1! 2^{b_1} b_1!}}{\sum_{i \text{ odd}} \frac{e^{-1}}{i!}} = \frac{2}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} - e^{-\frac{1}{2}})}.$$

For a_1 even, we get

$$\Pr[a_1, b_1 | a_1 \text{ even}] = \frac{\frac{e^{-\frac{3}{2}}}{a_1! 2^{b_1} b_1!}}{\sum_{i \text{ even}} \frac{e^{-1}}{i!}} = \frac{2}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} + e^{-\frac{1}{2}})}.$$

Hence, we get

$$\Pr[a_1, b_1] = \begin{cases} \frac{2\left(1 - \left(\frac{1+e^{-2}}{2}\right)^n\right)}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} - e^{-\frac{1}{2}})}, & \text{if } a_1 \text{ odd}, \\ \frac{2\left(\frac{1+e^{-2}}{2}\right)^n}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} + e^{-\frac{1}{2}})}, & \text{if } a_1 \text{ even}. \end{cases}$$

Now, we compute $\bar{N}_{n-1}$ as

$$\bar{N}_{n-1} = \sum_{\substack{a_1 \text{ even} \\ b_1}} N_{n-1}(a_1, b_1) \Pr[a_1, b_1 | a_1 \text{ even}] = \sum_{\substack{a_1 \text{ even} \\ b_1}} \frac{2N_{n-1}(a_1, b_1)}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} + e^{-\frac{1}{2}})}.$$

Afterwards, N_n by substituting $N_n(0, b_1) = 2b_1 + \bar{N}_{n-1}$ as

$$\begin{aligned} N_n &= \sum_{\substack{a_1 = 0 \\ b_1}} (2b_1 + \bar{N}_{n-1}) \Pr[0, b_1] + \sum_{\substack{a_1 > 0 \\ b_1}} N_n(a_1, b_1) \Pr[a_1, b_1] \\ &= \frac{2(\bar{N}_{n-1} + 1)}{(e + e^{-1})} \left(\frac{1 + e^{-2}}{2} \right)^n + \sum_{\substack{a_1 > 0 \text{ even} \\ b_1}} \frac{2N_n(a_1, b_1)}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} + e^{-\frac{1}{2}})} \left(\frac{1 + e^{-2}}{2} \right)^n \\ &+ \sum_{\substack{a_1 > 0 \text{ odd} \\ b_1}} \frac{2N_n(a_1, b_1)}{a_1! 2^{b_1} b_1! (e^{\frac{3}{2}} - e^{-\frac{1}{2}})} \left(1 - \left(\frac{1 + e^{-2}}{2} \right)^n \right). \end{aligned}$$

We compute some values of $\bar{N}_n$, N_n, and N_n^* in Table 3. As we see from this table, the number of iterations is upper bounded by 2 in any success case. Finally, the total complexity is

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

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

Table 3. Some values for $\bar{N}_n$, N_n, and N_n^*

n	1	2	3	4	5	6	7	8	9	10	50	100
$\bar{N}_n$	2.613	4.306	1.908	5.403	6.115	6.576	6.874	7.068	7.193	7.275	7.424	7.424
N_n	1.264	1.895	1.908	1.789	1.675	1.593	1.540	1.507	1.488	1.477	1.462	1.462
N_n^*	1.418	1.879	1.851	1.748	1.652	1.582	1.535	1.505	1.487	1.476	1.462	1.462

In the case of DES, we have $\ell_k = 56$ and $\ell_m = 64$. We take $n = 3$ to get $n^* \approx 1.865$ and $R_n \approx 2^{-7.338}$. 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 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 [2] proved that the best (non-related-key) generic attack in the ideal cipher model would require at least 2^{78} encryptions. This example shows that the result no longer holds in the related-key model.

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

Comparison with the Kelsey-Schneier-Wagner attack. Kelsey, Schneier, and Wagner presented a related-key attack against three-key triple encryption which has similar performances [9]. It consists in using

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

Then, $\mathsf{Enc}_K \circ \mathsf{Enc}^{-1}_{\varphi(K)} = C_{K_1} \circ C^{-1}_{K_1 \oplus \Delta}$ which only depends on K_1. Hence, 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. Contrary to ours, it has no extension to two-key triple encryption. However, this attack extends to the encrypt-encrypt-encrypt triple encryption mode whereas our attack is restricted to the encrypt-decrypt-encrypt construction.

Note that getting $\mathsf{Enc}_K \circ \mathsf{Enc}^{-1}_{\varphi(K)}(y) = z$ on a random y is equivalent to getting $\mathsf{Enc}_K(x) = y$ and $\mathsf{Enc}_{\varphi(K)}(x) = z$ 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 [12] 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 the three-key triple encryption using $r = 2$, $d = 2^{33}$ (known and chosen plaintexts), $t = 2^{88}/3$ triple encryptions, and $m = 2^{38}$ bits. This attack extends to the encrypt-encrypt-encrypt case and to the two-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.

3.3 Two-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

$$\mathsf{Enc}_K \circ \mathsf{Enc}^{-1}_{\varphi(K)} = \left(C_{K_1} \circ C^{-1}_{K_2}\right)^3 .$$

Fixed points of $\mathsf{Enc}_K \circ \mathsf{Enc}^{-1}_{\varphi(K)}$ are either the fixed points of $C_{K_1} \circ C^{-1}_{K_2}$ or the points in 3-cycles. We just proceed as in the previous attack. The probability to have a fixed points and b cycles of length 3 is $e^{-4/3}/a!3^b b!$ (Lemma 2), and, $E(a) = 1$ and $E(b) = 1/3$ (Lemma 3). The number of values x such that $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^3(x) = x$ is $a + 3b$. As in the previous attack, a fixed point of $\left(C_{K_1} \circ C^{-1}_{K_2}\right)^3$ is good if it is also a fixed point of $C_{K_1} \circ C^{-1}_{K_2}$. Notice that if the number of fixed points $(a + 3b)$ is not a multiple of 3, then we certainly have a good fixed point.

The final complexity is very similar to the three-key encryption case. The difference is that we no longer need an 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 $p = 63\%$.

Let a_i and b_i the number of 1-cycle and 3-cycles in the list L_i whose index i is in I. Similar to the previous attack, given a_1 good fixed points and b_1 bad ones of the first list, we have

$$N_n(a_1, b_1) = \frac{1}{\binom{a_1+3b_1}{a_1}} \sum_{i=1}^{3b_1+1} i \binom{a_1 + 3b_1 - i}{a_1 - 1} .$$

In addition, we observe that $N_n(0, b_1) = 3b_1 + \bar{N}_{n-1}$, where $\bar{N}_{n-1}$ denotes the expected number of iterations conditioned to that all a_i's are multiples of 3. As in the previous attack, we have

$$N_n = \sum_{a_1, b_1} N_n(a_1, b_1) \Pr[a_1, b_1] \quad \text{and} \quad N_n^* = \frac{N_n - ne^{-n}}{1 - e^{-n}},$$

and we write the joint probability $\Pr[a_1, b_1]$ as

$$\Pr[a_1, b_1] = \begin{cases} \dfrac{3\left(1 - \left(\frac{1+2e^{-\frac{3}{2}} \cos \frac{\sqrt{3}}{2}}{3}\right)^n\right)}{2a_1! 3^{b_1} b_1! \left(e^{\frac{4}{3}} - e^{-\frac{1}{6}} \cos \frac{\sqrt{3}}{2}\right)}, & \text{if } a_1 \text{ not a multiple of 3,} \\[2ex] \dfrac{e^{-\frac{4}{3}} \left(\frac{1+2e^{-\frac{3}{2}} \cos \frac{\sqrt{3}}{2}}{3}\right)^{n-1}}{a_1! 3^{b_1} b_1!}, & \text{if } a_1 \text{ a multiple of 3.} \end{cases}$$

Similarly, we compute $\bar{N}_{n-1}$ as

$$\bar{N}_{n-1} = \sum_{\substack{a_1 \text{ a multiple of 3} \\ b_1}} N_{n-1}(a_1, b_1) \Pr[a_1, b_1 | a_1 \text{ a multiple of 3 }].$$

By noticing that $N_{n-1}(0, b_1) = 3b_1 + \bar{N}_{n-2}$, we compute some values of $\bar{N}_n$, N_n, and N_n^* and obtain identical results with the previous attack. Therefore, the complexity of this attack is

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

While the first part of the algorithm works like in the three-key case with two variants in Fig. 1 and Fig. 2, the second part of the algorithm is shown in 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 all the 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.

Comparison with the Merkle-Hellman meet-in-the-middle attack. Merkle and Hellman proposed to use a simple collision algorithm to find collisions between

```
1: sort $I$ in increasing order with first the list of $i$'s with $\#L_i$ not a 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 $c_i$
12:        if $(K_1, K_2, K_1)$ consistent with data then
13:          yield $(K_1, K_2, K_1)$ and exit
14:        end if
15:      end for
16:    end for
17:  end for
18: end while
19: attack failed
```

Fig. 4. Attack on Two-Key Triple Encryption (Second Part)

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

	two-key				three-key			
cipher	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%
$C_{\mathsf{conservative}}$	2^{66}	2^{136}	2^{200}	2^{265}	2^{67}	2^{136}	2^{200}	2^{265}

the list of all $C_{K_2}^{-1}(0)$ and the list of all $C_{K_1}^{-1}(\mathsf{Enc}(C_{K_1}^{-1}(0)))$ [1]. 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 $C_{K_1}^{-1}(\mathsf{Enc}(C_{K_1}^{-1}(0))) = z$. We can look for z in the second dictionary. This doubles the memory complexity and increases the data complexity to essentially the entire codebook.

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).

Note that our attack can be extended to the encrypt-encrypt-encrypt case in a way that for $K = (K_1, K_2)$ and the relation $\varphi(K_1, K_2) = (K_2, K_1)$, we have

$$\mathsf{Enc}_K \circ \mathsf{Enc}_{\varphi(K)} = (C_{K_1} \circ C_{K_2})^3 ,$$

which allows us to attack against two-key triple encryption by using the similar way with the encrypt-decrypt-encrypt case.

4 Conclusion

We presented a new attack on triple encryption which uses related keys. It can use chosen messages or known plaintexts, but requires the entire codebook for related keys. Our attack is the best in the known plaintext attack category in the three-key case. Besides, the best attacks remain the Merkle-Hellman attack [1] in the two-key case and the Kelsey-Schneier-Wagner attack [9] in the three-key case. In addition to the attacks on triple encryption, we formalize the various ways to compare the complexity of related-key attacks and apply it to the recently proposed attacks on AES.

References

1. Ralph C. Merkle ands Martin E. Hellman. On the Security of Multiple Encryption. *Commun. ACM*, 24(7):465–467, 1981.
2. Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
3. Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptology*, 7(4):229–246, 1994.
4. Eli Biham. New Types of Cryptanalytic Attacks Using Related Keys (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 1994.

5. Eli Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.
6. Alex Biryukov and Dmitry Khovratovich. Related-Key Cryptanalysis of the Full AES-192 and AES-256. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009.*, volume 5912 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2009.
7. Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. Distinguisher and Related-Key Attack on the Full AES-256. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249. Springer, 2009.
8. Whitfield Diffie and Martin E. Hellman. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *Computer*, 10:74–84, 1977.
9. John Kelsey, Bruce Schneier, and David Wagner. Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 1996.
10. Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 196–208. Springer, 1993.
11. Stefan Lucks. Attacking Triple Encryption. In Serge Vaudenay, editor, *Fast Software Encryption, 5th International Workshop, FSE '98, Paris, France, March 23-25, 1998, Proceedings*, volume 1372 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 1998.
12. Raphael Chung-Wei Phan. Related-Key Attacks on Triple-DES and DESX Variants. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 15–24. Springer, 2004.
13. Paul C. van Oorschot and Michael J. Wiener. A Known Plaintext Attack on Two-Key Triple Encryption. In Ivan Damgård, editor, *Advances in Cryptology - EUROCRYPT '90, Workshop on the Theory and Application of of Cryptographic Techniques, Aarhus, Denmark, May 21-24, 1990, Proceedings*, volume 473 of *Lecture Notes in Computer Science*, pages 318–325. Springer, 1991.
14. Paul C. van Oorschot and Michael J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *J. Cryptology*, 12(1):1–28, 1999.
15. Serge Vaudenay. Related-key Attack against Triple Encryption based on Fixed points. In Javier Lopez and Pierangela Samarati, editors, *SECRYPT 2011 - Proceedings of the International Conference on Security and Cryptography, Seville, Spain, 18 - 21 July, 2011, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 59–67. SciTePress, 2011.
16. Michael J. Wiener. The Full Cost of Cryptanalytic Attacks. *J. Cryptology*, 17(2):105–124, 2004.