A Unified Approach to Related-Key Attacks

Eli Biham[1,⋆], Orr Dunkelman[2,⋆⋆], and Nathan Keller[3,⋆⋆⋆]

[1]Computer Science Department, Technion.
Haifa 32000, Israel
biham@cs.technion.ac.il
[2]ESAT/SCD-COSIC, Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
orr.dunkelman@esat.kuleuven.be
[3]Einstein Institute of Mathematics, Hebrew University.
Jerusalem 91904, Israel
nkeller@math.huji.ac.il

Abstract. This paper introduces a new framework and a generalization of the various flavors of related-key attacks. The new framework allows for combining all the previous related-key attacks into a complex, but much more powerful attack. The new attack is independent of the number of rounds of the cipher. This property holds even when the round functions of the cipher use different subkeys.
The strength of our new method is demonstrated by an attack on $4r$-round IDEA, for any r. This attack is the first attack on a widely deployed block cipher which is independent of the number of rounds. The variant of the attack with $r = 2$ is the first known attack on 8-round IDEA.

1 Introduction

In many applications the same block cipher is used with two unknown keys whose relation is known. To study the security of block ciphers in these situations the *Related-key attacks* framework was first presented in 1993 [3]. In a related-key attack, the attacker is allowed to ask for plaintexts encrypted under two (or more) related keys. This approach might seem unrealistic, as it assumes that the attacker can control some relations between the unknown keys. Still, there are some instances, e.g., the 2PKDP protocol [44], where this approach suggests practical attacks.

A block cipher susceptible to a related-key attack has some security concerns. It may not be suitable for other cryptographic primitives that use block ciphers as building blocks, e.g., hash functions. A famous example for this claim is the block cipher TEA [46]. A related-key property of TEA [33] was used in hacking Microsoft's Xbox architecture, which uses a Davies-Meyer hash function employing TEA as the underlying block cipher [47]. Another security concern is the fact that such a cipher cannot be used in protocols which allow key manipulation, such as the ones used in most inter-bank communications in the US which increment the key by one in each transaction. Sometimes, the security of the mode of operation of the block cipher is closely related to the immunity of the cipher to related-key attacks (as in the 3GPP case, as was shown in [29]).

⋆ This work was supported in part by the Israel MOD Research and Technology Unit.
⋆⋆ This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).
⋆⋆⋆ The research presented in this paper was supported by the Adams fellowship.

There are two classes of related-key attacks: The first class, originally presented by Biham [3] and independently by Knudsen [36], are attacks that use related-key plaintext pairs. These attacks use pairs of keys for which most of the encryption function is equivalent. Such relations exist when the key schedule is very simple. Also, in order for the attacks to succeed, the round functions have to be relatively weak (i.e., there exists a known plaintext attack on the round function given one or two input/output pairs). On the other hand, once such a relation can be found, it can be used to devise an attack on the cipher, where the attack is independent of the number of rounds.

The second class of related-key attacks, originally presented by Kelsey et al. [32, 33] is composed of attacks that treat the key relation as another freedom level in the examination of statistical properties of the cipher. Besides related-key differentials, where the key difference is used to control the evolution of differences, this class contains variants of most of the known cryptanalytic techniques: The SQUARE attack [20] was treated in the related-key model in [23] and used to extend the best known SQUARE attack against AES into a related-key attack that uses 256 related keys. The boomerang attack [45] and the rectangle attack [5] were combined with related-key differentials to introduce the related-key boomerang and related-key rectangle attacks [7, 28, 35]. Finally, linear cryptanalysis [38] was also combined with related-key attacks to produce a related-key attack on 7.5-round IDEA [8]. The second class of attacks can deal with much more complex key schedules and round functions, but their effectiveness (usually) drops with the number of rounds.

In this paper we unify the main ideas from the two classes of related-key attacks into one framework. The new framework has two main advantages:

1. A new approach for generating multiple related-key plaintext pairs, based on multiple encryptions under chains of related keys. When the key schedule has a short cycle, it is possible to obtain many related-key plaintext pairs from one pair using encryption under several keys. This technique allows to mount attacks on round functions that require more than two input/output pairs.
2. A combination of the two classes of attacks to allow a related-key attack on the underlying round function (rather than applying only a simple attack on the round function).

Thus, the unified approach allows attacking more ciphers, as the restrictions on the key schedule and on the round functions are significantly reduced.

To demonstrate the strength of the new technique, we apply our new attack to $4r$-round IDEA [37]. IDEA is a 64-bit block cipher with 128-bit keys, which was introduced by Lai and Massey in 1991. IDEA was thoroughly analyzed [1, 4, 7–9, 12, 16, 18, 19, 21, 22, 26, 27, 31, 39–41] but the best known attack on the cipher is against 7.5-round IDEA (out of 8.5 rounds) in the related-key model [8], and 6-round IDEA in the single key model [9].

We first introduce a related-key attack on 4-round IDEA. Then, using our new framework, we elevate this attack to any $4r$-round IDEA, presenting the first attack on IDEA that is independent of the number of rounds. The variant of the attack with $r = 2$ is the first known attack on 8-round IDEA.

The remainder of this paper is organized as follows: In Section 2 we present the various related-key attacks. We incorporate all the attacks into the new related-key framework in Section 3. Section 4 describes our attack on $4r$-round IDEA. Appendix A contains a short description of IDEA. Appendix B gives the full description of the 4-round related-key differential of IDEA we use in our attack. In Appendix C we outline a different attack algorithm on $4r$-round IDEA (with roughly the same data and time complexities). We conclude the paper in Section 5.

2 Previous Work

Related-key attacks exploit the relations between the encryption processes under different but related keys. Related-key attacks can be divided into two classes. The first class is attacks concentrated on detecting and exploiting related-key plaintext pairs, and the second class is adaptation of the standard cryptanalytic attacks into the related-key model.

2.1 Related-Key Attacks Exploiting Related-Key Plaintext Pairs

The original variant of related-key attacks introduced in [3, 36] is attacks exploiting related-key plaintext pairs. The main idea behind the attack is to find instances of keys for which the encryption processes deploy the same permutation (or almost the same permutation). To illustrate the technique, we shortly present the attack from [3].

Consider a variant of DES in which all the rotate left operations in the key schedule algorithm are by a fixed number of bits.[1] For any key K^1 there exists another key K^2 such that the round subkeys KR_i^1, KR_i^2 produced by K^1 and K^2, respectively, satisfy:

$$KR_{i+1}^1 = KR_i^2, \qquad \text{for } i = 1, \ldots, 15.$$

For such pair of keys, if a pair of plaintexts (P_1, P_2) satisfies $P_2 = f_{KR_1^1}(P_1)$, where $f_{sk}(P)$ denotes one round DES encryption of P under the subkey sk, then the corresponding ciphertexts C_1 and C_2 satisfy $C_2 = f_{KR_{16}^2}(C_1)$. Given such a pair of plaintexts (called in the sequel *a related-key plaintext pair*), the subkeys KR_{16}^2 and KR_1^1 can be easily extracted [3].

Throughout the paper we shall refer to the round in which the key is recovered as the *underlying round function.* This kind of related-key attacks is based on finding related-key plaintext pairs, which are then used to extract an input/output pair (or two pairs) to the round function. Once the input/output pair to the underlying round function is given, the attacker applies a cryptanalytic attack on the underlying round function and retrieves the key.

In the above attack, the attacker asks for the encryption of two pools of 2^{16} chosen plaintexts under two (unknown) related keys K^1, K^2. The plaintexts in the first pool, denoted by S_1, are of the form (X, A) and are encrypted under K^1, and the plaintexts in the second pool, denoted by S_2, are of the form (A, Y) and are encrypted under K^2, where A is some fixed 32-bit value and X, Y vary.

The attacker then finds pairs of ciphertexts (C_1, C_2), such that the first ciphertext belongs to S_1 and the second one belongs to S_2 and such that the left half of C_1 equals to the right half of C_2. Once such a pair is found, then there is a good chance that P_1 and P_2, the corresponding plaintexts, satisfy that $P_2 = f_{KR_1^1}(P_1)$. If this is the case, then the pair (P_1, P_2) is a related-key plaintext pair, and it can be used to retrieve the values of KR_{16}^2 and KR_1^1. It can be shown that with a high probability, if the pair is not a related-key plaintext pair, this procedure yields a contradiction, and hence, once a consistent value for KR_{16}^2 and KR_1^1 is suggested by the attack, it is the correct value with high probability.

The data complexity of the attack is 2^{17} chosen plaintexts, and the time complexity of the attack is 2^{17} encryptions as well. We note that even if there were more rounds in the modified version of DES, the attack would still be successful.

In the more general case, this class of related-key attacks is composed of three parts: Obtaining related-key plaintexts, identifying the related-key plaintext pairs, and using them to deduce the key. In many cases, identifying the related-key plaintext pairs is best achieved by assuming for each candidate pair that it is a related-key plaintext pair, and then using it as an

[1] Such a variant was proposed by Brown and Seberry [17].

input for the key recovery phase of the attack. In other cases, the round functions' weaknesses allow the attacker to identify these pairs easily.

The attack relies heavily on the simplicity of the key schedule, the similarity of the rounds, and on the cryptographic weakness of the underlying round function. As a result, most of the known block ciphers are immune to related-key attacks of this class.

2.2 Slide Attacks

When a cipher has self-related keys, i.e., it can be written as $E_k = f_k^\ell = f_k \circ f_k \circ \cdots \circ f_k$ it is susceptible to a variant of the related-key attack called the *slide attack* [13]. In this case, it is possible to apply the related-key attack to the cipher with $K^1 = K^2$, thus eliminating the key requirement of having two keys. The attacker looks for a slid pair, i.e., two plaintexts (P_1, P_2) such that $P_2 = f_k(P_1)$. In this case, the pair satisfies $C_2 = f_k(C_1)$ as well. When the round function f_k is simple enough, it is possible to use these two pairs in order to deduce information about the key.

In the slide attack the attacker obtains enough plaintext/ciphertext pairs to contain a slid pair, and has to check for each possible pair of plaintexts whether it is a slid pair by applying the attack on f_k. When dealing with a general block cipher this approach requires $O(2^{n/2})$ known plaintexts and $O(2^n)$ applications of the attack on f_k, where n is the block size. For Feistel block ciphers, the attack can be optimized using $O(2^{n/4})$ chosen plaintexts and $O(1)$ applications of the attack. Note that as in the original related-key attacks, the main drawback of this approach is that the attack can be used only if f_k can be broken using only two known input/output pairs, i.e., given one slid pair.

In 2000, Biryukov and Wagner [14] presented two variants of the slide attack, named *complementation slide* and *sliding with a twist*. These variants allow for treating more complex functions in the slide attack. Nevertheless, there are no widely used ciphers that can be attacked using these techniques.

The authors of [14] also presented several techniques aimed at finding several slid pairs simultaneously, enabling to use the attack even if several input/output pairs are needed for attacking f_k. One of these techniques, fully explored by Furuya [24], uses the fact that when (P_1, P_2) is a slid pair then $(E_k^t(P_1), E_k^t(P_2))$ are also slid pairs for all values of t.[2] This allows the attacker to transform any known plaintext attack on f_k that requires m known plaintexts to an attack on E_k with a data complexity of $O(m \cdot 2^{n/2})$ adaptively chosen plaintexts. The time complexity of this approach is $O(2^n)$ applications of the known plaintext attack on f_k.[3]

2.3 Attacks Adapting Standard Techniques to the Related-Key Model

The related-key model can be used as a platform for all standard attacks. This fact was first noted in [32, 33] where related-key differentials were introduced. Recall, that a regular differential deals with some plaintext difference ΔP and a ciphertext difference ΔC such that

$$\Pr_{P,K}[E_K(P) \oplus E_K(P \oplus \Delta P) = \Delta C]$$

is high enough. A related-key differential is a triplet of a plaintext difference ΔP, a ciphertext difference ΔC, and a key difference ΔK, such that

$$\Pr_{P,K}[E_K(P) \oplus E_{K \oplus \Delta K}(P \oplus \Delta P) = \Delta C]$$

[2] Throughout the paper the notation $F^i(\cdot)$ means i successive applications of $F(\cdot)$.

[3] It is worth mentioning that the technique can be easily improved in the case of Feistel ciphers, for which $O(2^{n/4})$ chosen plaintexts and $O(m)$ adaptive chosen plaintexts are sufficient to achieve m slid pairs, which are easily identified.

is high enough.

The related-key differential attack uses the subkey differences to control the development of differences through the encryption process. As a result, related-key differentials are usually much stronger than the respective "ordinary" differentials. The related-key differential technique was used to attack numerous block ciphers, including GOST, TEA, and 6-round KASUMI.

For example, using this approach a 60-round related-key differential with probability 2^{-30} of TEA is presented. Using this related-key differential, it is possible to break the full TEA (with 64 rounds) using about 2^{32} related-key chosen plaintexts and a small computational effort [33].

After the introduction of impossible differentials, i.e., differentials with zero probability, in [4], the concept of related-key impossible differentials was presented [30]. In this case, the subkey relations are used to ensure that the input difference of the impossible differential can not evolve into the output difference. This technique was used to analyze 8-round AES-192.

Related-key differentials were also used as the base for the related-key boomerang and the related-key rectangle attacks [7, 28, 35]. These attacks use two related-key differentials, i.e., up to four related keys. Hence, they enjoy the transition into related-key differentials twice, leading to much higher probabilities for the distinguisher (in exchange for more related keys). The related-key rectangle technique was successfully applied to several block ciphers, including 10-round AES-192, 10-round AES-256, the full SHACAL-1, the full KASUMI, and 7-round IDEA.

In [23] it is showed that the SQUARE attack can also be improved in the related-key model. The 4-round SQUARE property of AES used in the regular SQUARE attack, is extended into a 5-round related-key SQUARE property for AES-256. As a result, while the ordinary SQUARE technique can be used to attack up to 7 rounds of AES, the related-key SQUARE attack is applicable to a 9-round variant of AES.

Finally, even linear relations can be improved in the related-key model. In [8] a 2.5-round linear relation of IDEA is presented. When two related keys are used, this linear relation can be extended to a 4.5-round linear relation. This extension improves the regular attack on IDEA by 2.5 rounds, and is the best known attack so far against IDEA.

3 The Unified Related-Key Framework

In this section we present the new framework unifying the different related-key attacks. The construction of the framework is divided into two stages:

- First, we present a new approach to generating multiple related-key plaintext pairs, based on encryption under chains of related keys. This approach allows to mount a related-key attack on the entire cipher, even if the attack on the underlying function requires multiple input/output pairs.
- Then, we unify the two classes of related-key attacks into a single framework. This allows to use a related-key attack on the underlying round function. Thus, even if the underlying round function is secure against regular cryptanalytic attacks, it can still be attacked using a related-key attack.

3.1 A New Approach For Generating Multiple Related-Key Plaintext Pairs

When a cipher can be written as $E_k = f_k^\ell = f_k \circ f_k \circ \cdots \circ f_k$ and the slide attack can be applied, any slid pair (P, Q) can be used to generate many additional slid pairs of the form $(E_k^t(P), E_k^t(Q))$, for all t. These pairs can be used to devise a slide attack on the cipher even if multiple input/output pairs are required to break the underlying round function. However, this property exists only since the relation between the plaintexts of the slid pair is similar to the relation between the ciphertexts. If the plaintexts satisfy $Q = f_k(P)$ then the ciphertexts satisfy $E_k(Q) = f_k(E_k(P))$, and thus can be treated as the plaintexts in a new slid pair.

For an encryption with different subkeys, i.e., when $E_K = f_{k_r} \circ f_{k_{r-1}} \circ \cdots \circ f_{k_1}$, the situation is more complicated. The plaintexts of a related-key plaintext pair satisfy $Q = f_{k_1}(P)$, but the respective ciphertexts satisfy $E_k(Q) = f_{``k_{r+1}"}(E_k(P))$.[4] Hence, unless $k_1 = ``k_{r+1}"$, multiple encryption does not yield additional related-key plaintext pairs.

Our new approach uses chains of keys in order to achieve the additional related-key plaintext pairs. Let $E_K = f_{k_r} \circ f_{k_{r-1}} \circ \cdots \circ f_{k_1}$, and let (P^1, Q^1) be a related-key plaintext pair (with the corresponding ciphertexts (P^2, Q^2)) with respect to the keys (K_P^1, K_Q^1), i.e., $k_{i+1}^{P_1} = k_i^{Q_1}$, and $f_{k_1^{P_1}}(P^1) = Q^1$. Then, if K_P^2 is a key such that $k_1^{P_2} = k_r^{Q_1}$, then $f_{k_1^{P_2}}(P^2) = Q^2$. Moreover, let K_Q^2 satisfy that $k_{i+1}^{P_2} = k_i^{Q_2}$, then (P^2, Q^2) is a related-key plaintext pair with respect to (K_P^2, K_Q^2).

Defining K_P^2 as a function of K_P^1 is usually very simple, and usually it is the key that produces the next r subkeys if the key schedule would have been extended by r rounds. Formally, there are cases in which there exists a function $g(\cdot)$ such that for every pair of keys $(K_P^1, K_P^2 = g(K_P^1))$, and the key K_Q^1 related to K_P^1, we have $k_1^{P_2} = k_r^{Q_1}$. For example, in the modified variant of DES considered in Section 2, g can be the rotation of the key by 16 times the rotation in each round. Examples of real ciphers for which such g exists are $4r$-IDEA and the full SHACAL-1. In IDEA, where each f_{k_i} represents 4 rounds, we have $g(K) = K \lll (75 \cdot r)$. For SHACAL-1, $g(K)$ is obtained from K by running the LFSR used in the key schedule of the cipher 80 steps forward.

Assume now that for the examined cipher there exists a function g as described above. If the pair (P^1, Q^1) is a related-key plaintext pair with respect to the keys (K_P^1, K_Q^1), then the corresponding ciphertext pair (P^2, Q^2) is a related-key plaintext pair with respect to the keys $(g(K_P^1), K_Q^2)$, where K_Q^2 is the key related to $g(K_P^1)$ (and in many cases it is $g(K_Q^1)$).

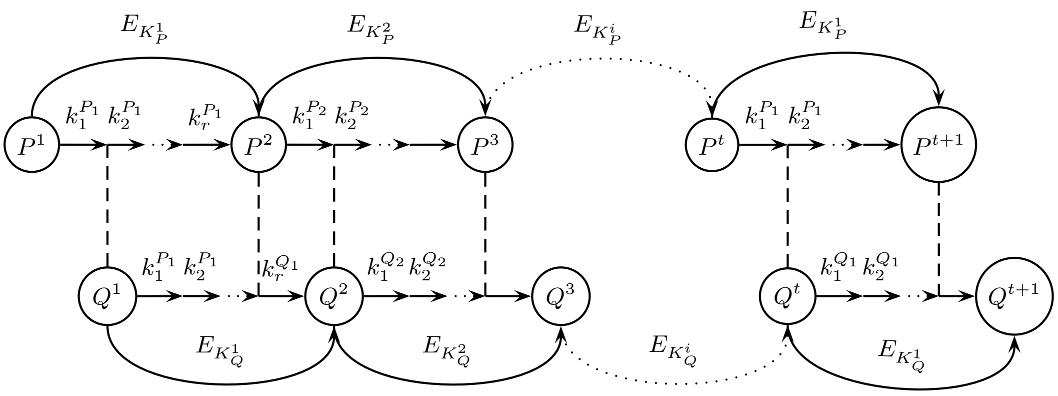
This process can be repeated to achieve multiple related-key plaintext pairs with respect to different pairs of related keys. We define $H_{K_P}^t = E_{g^{t-1}(K_P)} \circ E_{g^{t-2}(K_P)} \circ \ldots \circ E_{K_P}$ and similarly $H_{K_Q}^t = E_{g^{t-1}(K_Q)} \circ E_{g^{t-2}(K_Q)} \circ \ldots \circ E_{K_Q}$. If (P^1, Q^1) is a related-key plaintext pair with respect to (K_P, K_Q), then the pair $(H_{K_P}^t(P^1), H_{K_Q}^t(Q^1))$ is a related-key plaintext with respect to $(g^t(K_P), g^t(K_Q))$.

While in some cases obtaining many related-key plaintext pairs under different keys might be useful, we have not identified a concrete example where it can be used as is. We do note that for some specific cases this property can be used to identify the related-key plaintext pairs more easily. Assume that the related-key plaintext pair satisfies some relation in the ciphertexts which is not sufficient for the immediate identification of the related-key plaintext pair (for example, $n/4$ bits out of the n bits of the ciphertexts have to be equal). It is possible to identify the related-key plaintext pair by using the fact that a related-key plaintext pair is expanded into several such ones.

In most cases though, the attack on the underlying function requires several input/output pairs encrypted under the same key. However, we note that if for some t, $g^t(K_P^1) = K_P^1$, we can get more related-key plaintext pairs under the original key pair (K_P^1, K_Q^1). We outline the evolution of such a pair in Figure 1

For example, in $4r$-IDEA the cycle length of g for all the keys equals at most 64. Hence, using the algorithm presented above we can generate efficiently many related-key plaintext pairs

[4] "k_{r+1}" is the rth subkey produced by the second key. It can be treated as the $r+1$th subkey produced by the first key.



Dashed line stands for equal values.

Fig. 1. The Evolution of Multiple Related-Key Plaintext Pair

encrypted under the same pair of related keys. For block ciphers whose key schedule is based on LFSRs, e.g., SHACAL-1, the cycle size is $lcm(r, l)$ where l is the cycle length of the LFSR and r is the number of rounds of the cipher.

After obtaining enough related-key plaintext pairs under the keys (K_P, K_Q) it is possible to mount any known plaintext attack on the underlying round function, similarly to the slide case. If sufficiently many known plaintexts are available, then it might be possible to mount chosen plaintext attacks, or even adaptive chosen plaintext attacks. Therefore, the new approach for the generation of related-key plaintext pairs allows to mount the related-key attack even if the underlying function is not a weak one.

Our above observation can be used as-is, or in conjunction with the method we describe in the following section. We note that the data and time complexities required for the generation of the sequence are discussed separately, as they depend heavily on the structure of the analyzed cipher.[5]

3.2 A New Approach for Attacking the Underlying Round Functions

The method described in the previous section enables the attacker to produce many related-key plaintext pairs given one such pair. As noted earlier, these pairs can be used to mount any regular key recovery attack on the underlying round function. However, in many cases, no such attacks exist, while there is a related-key attack on the underlying round function, e.g., a related-key differential attack.

Our new framework allows to combine the related-key structure with a related-key attack on the underlying function. The main feature of the new framework is examining and comparing several chains of related-key plaintext pairs encrypted under different (but related) pairs of related keys.

Recall that $E_k = f_{k_r} \circ f_{k_{r-1}} \circ \cdots \circ f_{k_1}$, and assume that there exists a related-key attack on $f(\cdot)$. We shall describe the case of a related-key attack that requires two related keys, but related-key attacks which require more keys can be easily integrated into this framework.

Assume that the related-key attack on $f(\cdot)$ uses two related keys k_1 and $\hat{k_1}$. We denote the data used in the attack by the set of input/output pairs $(I_1, O_1), (I_2, O_2), \ldots$ encrypted under k_1, and the set of input/output pairs $(\hat{I_1}, \hat{O_1}), (\hat{I_2}, \hat{O_2}), \ldots$ encrypted under $\hat{k_1}$. There might

[5] For the sake of simplicity, we describe only attacks that use pairs encrypted under the same pair of related keys. In some cases, the attack can be improved by aggregating the information obtained from several pools of related-key plaintext pairs encrypted under different pairs of related keys.

be some relation between the inputs, e.g., in the case of a related-key differential attack, the relation between the inputs is $I_j \oplus \hat{I}_j = \Delta_{IN}$.

Our unified attack allows to mount the attack on $f_{k_1}(\cdot)$ although the outputs of $f_{k_1}(\cdot)$ are not immediately available to the attacker. The outputs are detected as the related-key plaintext counterparts of the inputs (if more than one input/output pair is needed under a single pair of keys, they can be generated using the approach provided in Section 3.1).

The basic algorithm of the unified attack is the following:

1. Pick two plaintexts P and R. For every possible pair of plaintexts Q and S, perform the following:
   (a) Assume that (P, Q) is a related-key plaintext pair with respect to the keys (K_P, K_Q) and generate from them a chain of related-key plaintext pairs (P^t, Q^t) with respect to the same pair of keys.
   (b) Assume that (R, S) is a related-key plaintext pair with respect to the keys (K_R, K_S) and generate from them a chain of related-key plaintext pairs (R^m, S^m) with respect to the same pair of keys.
   (c) Detect a set of (P^{t_i}, Q^{t_i}) such that $P^{t_i} = I_i$, and let $O_i = Q^{t_i}$.
   (d) Detect a set of (R^{m_j}, S^{m_j}) such that $R^{m_j} = \hat{I}_j$, and let $\hat{O}_j = S^{m_j}$.
   (e) Apply the related-key attack on $f_{k_1}(\cdot)$ and $f_{\hat{k_1}}(\cdot)$ using the inputs $I_1, I_2, \ldots$ and $\hat{I}_1, \hat{I}_2, \ldots$ and the corresponding outputs $O_1, O_2, \ldots$ and $\hat{O}_1, \hat{O}_2, \ldots$.
2. If for all the checked pairs Q and S the related-key attack on f_{k_1} fails, repeat Step 1 with a different choice of P and R.

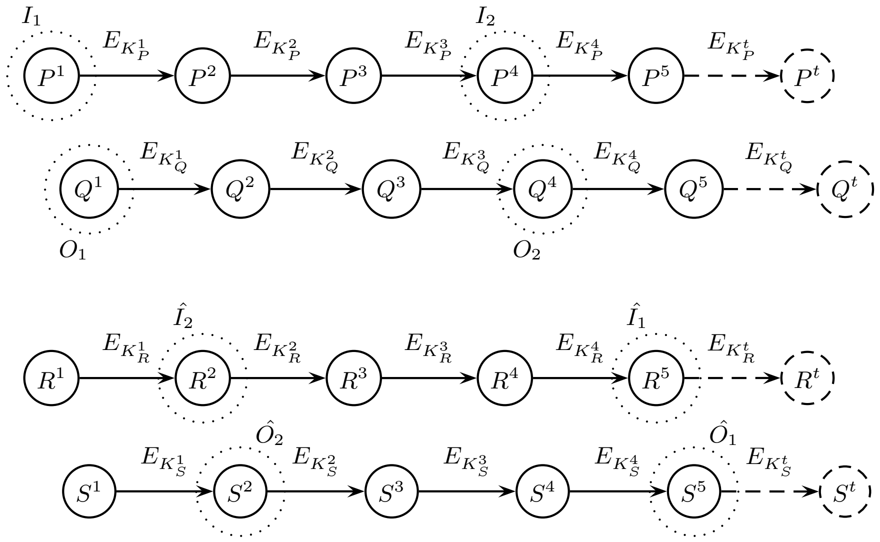
The unified attack considers simultaneously two chains $\{(P^t, Q^t)\}$ and $\{(R^m, S^m)\}$ encrypted under different pairs of related keys. Each chain contains a set of input/output pairs for the round functions f_{k_1} and $f_{\hat{k_1}}$, thus allowing to apply the related-key attack. We outline these chains of plaintexts and their relations in Figure 2.

Our new approach increases the problem of identifying the related-key plaintext pairs. Now, the attacker has to find two related-key plaintext pairs (P, Q) and (R, S) rather than only one.

The general algorithm can be improved in many cases. The relations between I_i's and O_i's can be used to reduce the number of related-key plaintext counterparts corresponding to P and (independently) to R [3, 13]. For example, if $f(\cdot)$ is one round of a Feistel cipher, then the number of possible counterparts of P (and of R) is greatly reduced. If there exists a relation between the values of O_i's and the values of $\hat{O}_j$'s they can be used as well to reduce the need of trying all possible pairs of pairs ((P, Q) and (R, S)). We observe that the possible number of counterparts can further be reduced using relations between the chains. For example, assume that there exists a related-key differential of $f(\cdot)$ that predicts that with high probability the input difference α becomes an output difference β. In this case, obtaining the first related-key plaintext pair (P, Q) suggests that with a high probability $(R = P \oplus \alpha, S = Q \oplus \beta)$ is also a related-key plaintext pair. We note that in the case of the slide attack, a similar improvement is suggested in [14].

3.3 Comparison with Other Related-Key Attacks

The main drawback of our proposed framework is the fact that the new attack requires encryption under multiple related keys. Hence, in order to measure the effectiveness of the new framework, it is not sufficient to compare it with the classic generic attacks, such as exhaustive key search and dictionary attacks. The framework should be compared also to generic attacks that allow the attacker to use encryption under multiple related keys. In this section we consider two attacks of this class.



(P^i, Q^i) are related-key plaintext pairs with respect to (K_P^i, K_Q^i).
(R^j, S^j) are related-key plaintext pairs with respect to (K_R^j, K_S^j).
(I_i, O_i) are input/output pairs for the related-key attack on $f(\cdot)$ for the first key.
$(\hat{I}_i, \hat{O}_i)$ are input/output pairs for the related-key attack on $f(\cdot)$ for the second key.

Fig. 2. Overview of the Related-Key Plaintext Pairs Used in the Unified Attack

The first attack is a generic time-memory-key trade-off attack suggested in [11]. In the classic time-memory trade-off attack on block ciphers, if the number of keys is N, the available memory is M, and the time complexity of the on-line step of the attack is T, then $N^2 = TM^2$. In addition, the attack requires a precomputation step of N operations. In [11] the authors show that if the attacker is able to ask for encryptions under D related keys, then the complexities of the attack can be reduced according to the curve $(N/D)^2 = TM^2$. The length of the precomputation step is also reduced to N/D.

In view of this generic attack, it seems that an attack requiring encryption under D related keys should be compared to an exhaustive search over a space of N/D keys or to a classic time-memory tradeoff attack over such key space.[6]

The second attack is the generic attack presented in [2]. The attack uses the fact that if for any block cipher a key is periodic, i.e., can be rotated to itself, this can be identified easily using a few related-key queries. Actually, this property defines a weak key class for all block ciphers. In the attack, the attacker asks for the encryption under various keys, tracing the relation of the keys to the original key, and checks whether the related keys fall into the weak key class.

We note that the attack can be applied with other weak key classes as well. In general, if the size of a weak key class is WK and the total number of possible keys is N, the attack is

[6] We note that the time-memory-key attack recovers only one of the related keys. However, the other keys can be easily found using the relations between the related keys. Also note that if the relation between the keys is correlated to the tables constructed in the time-memory-key attack, the attack might fail.

expected to require N/WK related keys. For the generic weak-key class presented in [2], the attack requires $2^{N/2}$ related keys and a few chosen plaintext queries under each of the keys.

The attack presented in Section 4.2 requires 256 related keys, has a memory complexity of 2^{66} and a time complexity of 2^{100} for a 128-bit key cipher. Hence, its complexity is better than that of the corresponding time-memory-key tradeoff attack. It also compares favorably with the generic attack presented in [2] since for such a small amount of related keys the success probability of the generic attack is 2^{-56}.

4 Attacking $4r$-Round IDEA

IDEA is a 64-bit, 8.5-round block cipher with 128-bit keys [37]. IDEA is a composition of XOR operations, additions modulo 2^{16}, and multiplications over the field $GF(2^{16}+1)$. The full description of IDEA is given in Appendix A.

4.1 Observations on IDEA Used in the Attack

Our attack on IDEA is based on the following two observations:

1. The key schedule of IDEA has the following property: If the original key is rotated by 75 bits to the left and entered into the key schedule algorithm, the resulting subkeys of rounds 1–4 are the same as the subkeys of rounds 5–8 for the original key. Hence, we can treat $4r$-IDEA as a cascade of r 4-round IDEA components.
2. There exists a related-key truncated differential on 4-round IDEA. The key difference of the differential is in bits 25 and 48. The input difference of the differential is $\Delta_{IN} = (0, 8040_x, 0, 0)$ and the fourth input word is set to 1. This input difference leads to an output difference $\Delta_{OUT} = (a, a, b, b)$, where a and b are some (not necessarily different) 16-bit values, with probability of 2^{-17}. The related-key truncated differential is fully described in Appendix B.

We denote four rounds of IDEA with key k (i.e., the first 128 bits that are used as subkeys), by $4IDEA_k$. Thus, a $4r$-round IDEA with a key K can be described as

$$E_K(P) = 4IDEA_{K \lll 75\cdot(r-1)}(\ldots(4IDEA_{K\lll 75}(4IDEA_K(P)))),$$

where $\lll$ is the rotate left operation. The attack on $4r$-round IDEA uses the unified related-key framework. The cipher is treated as a cascade of r 4-round components, and the related-key truncated differential is used to attack the underlying function, i.e., 4-round IDEA.

A pair of ciphertexts $C^1 = (C_1^1, C_2^1, C_3^1, C_4^1)$ and $C^2 = (C_1^2, C_2^2, C_3^2, C_4^2)$ that satisfy the output difference Δ_{OUT} satisfy that

$$C_1^1 \oplus C_1^2 = C_2^1 \oplus C_2^2 \quad \text{and} \quad C_3^1 \oplus C_3^2 = C_4^1 \oplus C_4^2$$

These relations can be easily rewritten into:

$$C_1^1 \oplus C_2^1 = C_1^2 \oplus C_2^2 \quad \text{and} \quad C_3^1 \oplus C_4^1 = C_3^2 \oplus C_4^2$$

Thus, we define the function $evaluate(C)$, to efficiently help us to determine right pairs:

$$evalute(C = (C_1, C_2, C_3, C_4)) = C_1 \oplus C_2 || C_3 \oplus C_4.$$

Thus, in order to check whether two values (C^1, C^2) satisfy the output difference of the related-key differential, it is sufficient to check whether $evaluate(C^1) = evaluate(C^2)$.

For the sake of simplicity we shall describe the attack on 8-round IDEA. The changes needed for attacking $4r$-round IDEA with $r \neq 2$ are relatively small (and are mainly in the data generation phase).

The attack has three main steps. The first one is data generation, where chains of plaintexts are generated according to the framework we described in the previous section. The purpose of the chains is to produce several related-key plaintext pairs simultaneously. The second step is composed of analyzing the chains and trying to find the related-key plaintext pairs efficiently. This step is performed by using the key recovery step (the third one). The last step is the key recovery step, in which the candidates for being related-key plaintext pairs are used for key recovery. Once sufficiently many related-key plaintext pairs are found, so does the right key.

For sake of simplicity we assume that the chains of plaintexts generated in the attack compose the entire code book, i.e., all plaintexts are there. The cases when this assumption does not hold are discussed in Section 4.4. As long as the chains contain enough related-key plaintext pairs, our attack works. To justify the assumption we made, we note that starting with a chain which does not contain enough plaintexts is highly unlikely.

4.2 The Attack Algorithm

1. **Data Generation:**
   (a) Let K_P be the unknown key, and let $K_Q = K_P \lll 75$. Let $K_R = K_P \oplus e_{25,48}$, i.e., K_R is the same as K_P in all bits but bits 25 and 48. Finally, let $K_S = K_R \lll 75$. Pick randomly four plaintexts: $P_0^0, Q_0^{75}, R_0^0, S_0^{75}$.
   (b) Starting from P_0^0 and K_P compute the following chain:

$$P_i^{l+22 \bmod 128} = \begin{cases} E_{K_P \lll l}(P_i^\ell) & \text{if } l+22 \not\equiv 0 \bmod 128 \\ E_{K_P \lll l}(P_{i-1}^\ell) & \text{if } l+22 \equiv 0 \bmod 128 \end{cases}$$

   Continue till P_0^0 is about to be encrypted under K_P again (according to our assumption — after 2^{64} encryptions under K_P). We denote this chain by $Chain_P$.
   (c) Compute $Chain_Q$ starting from Q_0^{75} as the plaintext and K_Q as the key, using the same process.
   (d) Denote by $K_R = K_P \oplus e_{25,48}$, Pick a plaintext R_0^0 randomly, and repeat the previous step with the key K_R to obtain the chain $Chain_R = R_0^0, R_0^{22}, \ldots, R_{2^{64}-1}^{106}$.
   (e) Pick a plaintext S_0^{75} randomly and perform the same operation in the previous step with the key $K_S = K_R \lll 75$, obtaining the chain $Chain_S = S_0^{75}, S_0^{97}, \ldots, S_1^{75}, \ldots, S_{2^{64}-1}^{53}$.
   For sake of simplicity, we assume that each of the four chains covers all possible plaintexts (i.e., each plaintext is encrypted under every key of the chain). We deal with the case of several chains in Section 4.4.
2. **Analyzing the Chains:** Locate a set of 2^{36} pairs of plaintexts $(P_{i_1}^0, P_{i_2}^0)$ in $Chain_P$ whose fourth word equals 1 for both plaintexts. For each such pair:
   (a) Compute the values of j_1 and j_2, such that $R_{j_1}^0 = P_{i_1}^0 \oplus \Delta_{IN}$ and $R_{j_2}^0 = P_{i_2}^0 \oplus \Delta_{IN}$.
   (b) For each $S_m^{75} \in Chain_S$ store the 64-bit value $value_S = evaluate(S_m^{75})||evaluate(S_{m+j_2-j_1}^{75})$ along with m in a table $Table_S$ indexed by the computed value.
   (c) For each $Q_l^{75} \in Chain_Q$ perform:
      – Compute the 64-bit value $value_Q = evaluate(Q_l^{75})||evaluate(Q_{l+i_2-i_1}^{75})$. Search for $value_Q$ in $Table_S$.
      – For each possible value of m associated with $value_Q$, check whether

$$Q_l^{75} = 4IDEA_{K_P}(P_{i_1}^0); \qquad Q_{l+i_2-i_1}^{75} = 4IDEA_{K_P}(P_{i_2}^0);$$

$$S^{75}_m = 4IDEA_{K_R}(R^0_{j_1}); \qquad S^{75}_{m+j_2-j_1} = 4IDEA_{K_R}(R^0_{j_2})$$

using the key recovery attack that uses the respective pairs (we outline this attack later), where 4IDEA denotes 4-round IDEA. If the key recovery attack succeeds, the key is found.

$Chain_P$ and $Chain_Q$ contains input/output pairs to $4IDEA_{K_P}(\cdot)$ whose order is unknown. The same is true for $Chain_R$ and $Chain_S$ with respect to $4IDEA_{K_R}(\cdot)$. The attack tries all the possible shifts between $Chain_P$ and $Chain_Q$ (for which one is the correct shift). To prevent the need of checking all the shifts between $Chain_R$ and $Chain_S$, we use the related-key truncated differential. The key difference between K_P and K_R is the key difference of the differential, which means that an input pair $(P^0_i, R^0_j = P^0_i \oplus \Delta_{IN})$ is more likely to have the corresponding outputs (Q^{75}_l, S^{75}_m) satisfying the output difference of the differential.

The attack algorithm first tries to find the shift between $Chain_P$ and $Chain_Q$. For each such shift, we assume it is the correct one and we find 2^{36} pairs of pairs $(P^0_{i_1}, R^0_{j_1})$ and $(P^0_{i_2}, R^0_{j_2})$ with difference Δ_{IN}. If indeed the shift was correct, the probability that the corresponding outputs satisfy the output difference (twice) is 2^{-34}, and thus, the only remaining problem is finding the corresponding outputs. For the $Chain_P$, as we know the shift of $Chain_Q$ we know the outputs. Thus, we only need to find the shift of $Chain_S$ with respect to $Chain_R$.

The last task is achieved by observing that if S^{75}_m is the output of R_{j_1}, then $S^{75}_{m+j_2-j_1}$ is the output of R_{j_2}. Thus, we compute for each S^{75}_m the value of $evaluate(S^{75}_m)||evalaute(S^{75}_{m+j_2-j_1})$. Similarly, if Q^{75}_l is the output of $P^0_{i_1}$, then $Q^{75}_{l+i_2-i_1}$ is the output of $P^0_{i_2}$. If indeed the pairs $(P^0_{i_1}, R^0_{j_1})$ and $(P^0_{i_2}, R^0_{j_2})$ satisfy the differential, then it must hold that $evaluate(S^{75}_m) = evalute(Q^{75}_l)$ and that $evalaute(S^{75}_{m+j_2-j_1}) = evalaute(Q^{75}_{l+i_2-i_1})$. Thus, the attack succeeds in retrieving the right shift of $Chain_S$ with respect to $Chain_R$ given the right shift of $Chain_Q$ with respect to $Chain_P$.

The most basic attack retrieves the subkey bits involved in the last MA layer (by finding the key value for which both pairs have a zero difference before the MA layer). An additional fast filtering of wrong subkey values can be performed using the first round of the truncated differential (verifying that indeed during the first KA layer the difference in the second word becomes 8000_x).

If indeed the related-key plaintext pairs are the ones analyzed, then there is a probability of 2^{-34} that the attack succeeds (as both pairs should be right pairs for the attack to succeed). If this is not the case, then it is highly unlikely that the key recovery attack succeeds. We first note that for any value of Q^{75}_l we expect about one suggestion for the value of m. Then, the probability that two wrong pairs (or even one wrong pair and one right pair) agree on the key for the fourth round MA layer is 2^{-32}. Considering the additional filtering based on the first round as well, the probability that a wrong Q^{75}_l leads to a consistent key suggestion is 2^{-34}.

Thus, it is expected that of the 2^{64} possible relations for a given $P^0_{i_1}$ and $P^0_{i_2}$, only 2^{30} values of Q^{75}_l may seem suitable, and these can be easily discarded using an additional "pair" of related-key plaintexts and ciphertexts.

4.3 The Time Complexity of the Attack

The first step of the attack is composed of constructing four chains. Each such chain requires the encryption of 2^{64} values, each under 64 keys. Thus, the time complexity of the data generation is $4 \cdot 64 \cdot 2^{64} = 2^{72}$ encryptions.

The time complexity of Step 2 is mostly dominated by Step 2(c). It is easy to see that using a well chosen data structure, Step 2(a) takes a relatively small number of operations for each pair. For each of the 2^{36} pairs $(P^0_{i_1}, P^0_{i_2})$, the time complexity of Step 2(b) is about 2^{64}

operations. Step 2(c) is repeating 2^{64} times a basic operation and a key recovery attack. The key recovery attack can be efficiently simulated using two table lookups (to a table suggesting for each pair the respective MA layer subkey). Thus, the time complexity of Step 2 is about $2^{36+64} = 2^{100}$ operations.[7]

We conclude that the attack requires 2^{72} related-key chosen plaintexts, and has a time complexity of about 2^{100} operations.

The memory complexity of the attack is dominated by the stored data and the table containing $value_S$. Thus, the total memory required by our attack is at most $4 \cdot 2^{64}$ blocks of memory for the data, and additional 2^{64} entries of the form $m||value_S$ (which take two blocks of memory each). Thus, the total memory used by the attack is at most $48 \cdot 2^{64}$ bytes (which are $6 \cdot 2^{64}$ blocks of 64 bits).

4.4 Dealing with Multiple Chains

When the chains do not cover the entire plaintext space, then the attack algorithm has to be tweaked a bit to ensure success. We first note that we can divide the entire plaintext space into multiple chains, i.e., $Chain_{P_1}$, $Chain_{P_2}$, ..., and equivalent $Chain_{Q_1}$, $Chain_{Q_2}$, ... Then, we wait till 2^{36} pairs of values $(P^0_{i_1}, P^0_{i_2})$ that can be used in the attack are encountered in some chain $Chain_{P_n}$.

We can treat the chains as generated by a random permutation over the plaintext space of a given key, e.g., K_P. Hence, the analysis of [25] can be applied, revealing that the longest chain is expected to cover about 2^{63} plaintexts (for a given key), that the second longest chain covers about 2^{62} plaintexts, etc.

As in the original description, we start from a random plaintext and start generating the chain. With high probability this plaintext belongs to one of the longest chains. Doing the same for the generation of the other chains, we are expected to find chains of the same length, i.e., $Chain_{P_i}$ and $Chain_{Q_i}$ of the same length, and $Chain_{R_j}$ and $Chain_{S_j}$ of the same length. As the existence of a related key plaintext pair in the two chains $Chain_{P_i}$ and $Chain_{Q_i}$ can happen only if their length is the same, then this can improve the attack, by first generating the chains, and then reducing the number of candidate related-key plaintext pairs by taking the chain lengths into account.

There is a small problem that may rise. In order for the attack to work, $(P^0_{i_1}, P^0_{i_2})$ counterparts, i.e., $R^0_{j_1}$ and $R^0_{j_2}$ has to be in the same chain $chainR_j$ (for the indexing phase done in Step 2(b)). This condition cannot be assured. However, assuming that the cycle structure of the chains $Chain_{R_1}$, $Chain_{R_2}$, ... behaves as if 8-round IDEA is a random permutation, it is expected that there is a chain $Chain_R$ whose size is larger than 2^{63} with overwhelming probability [25]. If this is the case, we can change the attack algorithm such that only pairs of plaintexts $(P^0_{i_1}, P^0_{i_2})$ whose counterparts are in that chain, are considered. This increases the number of pairs that are considered from 2^{36} to 2^{38} at most, but as at least 3/4 of the new pairs are not analyzed, this does not increase the time complexity of the attack.

Actually, the time complexity of the attack is expected to drop by a factor of about 2. This is caused by the fact that the indexing is now performed in some $Chain_S$ (the one corresponding to $Chain_R$) whose size is smaller than 2^{64}. Also, there are less candidates for Q^{75}_l that need to be considered.

We note that we can use even shorter chains, as long as there are sufficiently many candidate pairs between the chains $Chain_{P_i}$ and $Chain_{R_j}$. hence, the attacker starts to generate $Chain_{P_i}$

[7] By generating the indices using a little different order, it is possible to use eight 64-bit logical operations for the computation of $value_Q$ or $value_S$. Thus, the term "operation" refers here to about eighteen 64-bit logical operations and seven memory accesses.

until he obtains the three longest chains. Then, the attacker has to find a $Chain_Q$ with any of these lengths (which can be easily done, as the probability of picking a plaintext at random from the longer chains is significantly higher). The same is done for $Chain_{R_j}$ and $Chain_S$, where we stop once we have sufficiently long chains (and thus, enough candidate pairs for the analysis).

Thus, in this case our attack requires about 2^{99} 8-round IDEA encryptions. The data complexity can be slightly reduced as well. For generating $Chain_{P_i}$ and $Chain_{R_j}$ we do not need to cover all the small chains. Hence, it is expected that generating these chains requires roughly $2^{64} - 2^{52}$ adaptive chosen plaintexts and ciphertexts encrypted under 64 keys each. For $Chain_{Q_i}$ and $Chain_{S_j}$ we need only to find one of the longer chains, and thus it is expected that $2^{63.4}$ adaptively chosen plaintexts and ciphertexts (encrypted under 64 keys each) are required. Thus, the data complexity of the attack is $2 \cdot 2^{70} + 2 \cdot 2^{69.4} = 2^{71.7}$ related-key adaptive chosen plaintexts and ciphertexts.

We note that this attack requires more data than the entire codebook for a given key. However, for any given key of the 256 involved keys, we do not require the entire code book.

4.5 Changes for $4r$-Round IDEA with $r \neq 2$

As mentioned earlier, it is possible to apply our attack to $4r$-IDEA when the number of rounds is much larger than 8. The changes in the attack algorithm are only in the data generation phase. While for 8-round IDEA, the chains are constructed as $Chain_P = P_0^0, P_0^{22}, \ldots$, for $4r$-round IDEA the chains are of the form $Chain_P = P_0^0, P_0^{75 \cdot r \bmod 128}, P_0^{75 \cdot r \cdot 2 \bmod 128}, P_0^{75 \cdot r \cdot 3 \bmod 128}, \ldots$.

When $gcd(r, 128) = 2$, the obtained attack has the same data complexity as well as the same number of involved keys (64 for each of the chains). Besides the data generation phase, the attack is the same.

For the cases when $gcd(r, 128) = 1$ the chains $Chain_P$ and $Chain_Q$ are actually the same chain (and $Chain_R$ and $Chain_S$ as well). This follows the fact that for such values of r, K_P is rotated each time to the left by a number of bits which is eventually equal to 75 (i.e., there exists g s.t. $75 \cdot r \cdot g \equiv 75 \bmod 128$). Again, this has no affect on any other steps of the attack or on its date complexity.

When $gcd(r, 128) > 2$, the all the chains are shorter, as the number of keys needed for closing the cycle of the key schedule algorithm is shorter than 64. In that case, the data complexity of the attack drops by a factor of $gcd(r, 128)/2$, as well as the number of keys. For example, for $r = 128$, the data complexity of the attack is only 2^{66} plaintexts, and the number of keys is reduced to four.

5 Summary and Conclusions

Our new framework combines the various kinds of related-key attacks. We show that by combining them, we create a powerful attack. For example, we present the first attack on $4r$-round IDEA, and in particular, the first published work that can break 8 rounds of IDEA. The complexities of our new attack on IDEA, along with the best previously known attacks, are summarized in Table 1.

References

1. Eyup S. Ayaz, Ali A. Selçuk, *Improved DST Cryptanalysis of IDEA*, proceedings of Selected Areas in Cryptography 2006, Lecture Notes in Computer Science 4356, pp. 1–14, Springer-Verlag, 2007.

Rounds	Attack Type	Complexity			# of Keys	Source
		Data	Time	Memory		
5	Differential-Linear	16 KP	2^{114}	32	1	[9]
5.5	Higher-Order Diff.-Lin.	2^{34} CP	$2^{126.8}$	2^{35}	1	[9]
6	Higher-Order Diff.-Lin.	$2^{64} - 2^{52}$ KP	$2^{126.8}$	2^{64}	1	[9]
7	Related-Key Rectangle	2^{65} RK-CP	$2^{104.2}$	2^{66}	4	[8]
7.5	Related-Key Linear	$2^{43.5}$ RK-KP	$2^{115.1}$	$2^{44.5}$	2	[8]
8	Unified Related-Key	2^{72} RK-KP	2^{100}	$2^{66.6}$	256	Section 4
8	Unified Related-Key	$2^{71.7}$ RK-ACP	2^{99}	$2^{66.6}$	256	Section 4.4
8	Unified Related-Key	2^{71} RK-ACP	2^{103} MA	$2^{66.6}$	256	Appendix C
$4r$	Unified Related-Key	$2^{72}/x$ RK-KP	2^{100}	$2^{66.6}$	$256/x$	Section 4

$x = \lceil (gcd(r, 128)/2 \rceil$

KP – Known plaintext, CP – Chosen plaintext, RK – Related key,
ACP – Adaptive chosen plaintexts, MA – Memory accesses.
Time complexity is measured in encryption units (unless stated otherwise).
Memory complexity is measured in blocks.

Table 1. Selected Known Attacks on IDEA and Our New Results

2. Mihir Bellare, Tadayoshi Kohno, *A Theoretical Treatment of Related-Key Attacks: RKA-PRPs, RKA-PRFs, and Applications*, Advances in Cryptology, proceedings of EUROCRYPT 2003, Lecture Notes in Computer Science 2656, pp. 491–506, Springer-Verlag, 2003.
3. Eli Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, Journal of Cryptology, Vol. 7, No. 4, pp. 229–246, Springer-Verlag, 1994.
4. Eli Biham, Alex Biryukov, Adi Shamir, *Miss in the Middle Attacks on IDEA and Khufu*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 124–138, Springer-Verlag, 1999.
5. Eli Biham, Orr Dunkelman, Nathan Keller, *The Rectangle Attack – Rectangling the Serpent*, Advances in Cryptology, proceedings of EUROCRYPT 2001, Lecture Notes in Computer Science 2045, pp. 340–357, Springer-Verlag, 2001.
6. Eli Biham, Orr Dunkelman, Nathan Keller, *New Combined Attacks on Block Ciphers*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 126–144, Springer-Verlag, 2005.
7. Eli Biham, Orr Dunkelman, Nathan Keller, *Related-Key Boomerang and Rectangle Attacks*, Advances in Cryptology, proceedings of EUROCRYPT 2005, Lecture Notes in Computer Science 3494, pp. 507-525, Springer-Verlag, 2005.
8. Eli Biham, Orr Dunkelman, Nathan Keller, *New Cryptanalytic Results on IDEA*, accepted to Advances in Cryptology, proceedings of ASIACRYPT 2006, Lecture Notes in Computer Science 4284, pp. 412–427, Springer-Verlag, 2006.
9. Eli Biham, Orr Dunkelman, Nathan Keller, *A New Attack on 6-Round IDEA*, Fast Software Encryption 14, Lecture Notes in Computer Science 4593, pp. 211–224, Springer-Verlag, 2007.
10. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
11. Alex Biryukov, Sourav Mukhopadhyay, Palash Sarkar, *Improved Time-Memory Trade-Offs with Multiple Data*, proceedings of Selected Areas in Cryptography 2005, Lecture Notes in Computer Science 3897, pp. 110-127, Springer-Verlag, 2006.
12. Alex Biryukov, Jorge Nakahara Jr., Bart Preneel, Joos Vandewalle, *New Weak-Key Classes of IDEA*, proceedings of Information and Communications Security 4, Lecture Notes in Computer Science 2513, pp. 315–326, Springer-Verlag, 2002.
13. Alex Biryukov, David Wagner, *Slide Attacks*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 245–259, Springer-Verlag, 1999.

14. Alex Biryukov, David Wagner, *Advanced Slide Attacks*, Advances in Cryptology, proceedings of EUROCRYPT 2000, Lecture Notes in Computer Science 1807, pp. 586–606, Springer-Verlag, 2000.
15. Nikita Borisov, Monica Chew, Robert Johnson, David Wagner, *Multiplicative Differentials*, proceedings of Fast Software Encryption 9, Lecture Notes in Computer Science 2365, pp. 17–33, Springer-Verlag, 2002.
16. Johan Borst, Lars R. Knudsen, Vincent Rijmen, *Two Attacks on Reduced Round IDEA*, Advances in Cryptology, proceedings of EUROCRYPT '97, Lecture Notes in Computer Science 1233, pp. 1–13, Springer-Verlag, 1997.
17. Lawrence Brown, Jennifer Seberry, *Key Scheduling in DES Type Cryptosystems*, proceedings of AUSCRYPT 1990, Lecture Notes in Computer Science 453, pp. 221–228, Springer-Verlag, 1990.
18. Joan Daemen, René Govaerts, Joos Vandewalle, *Cryptanalysis of 2.5 Rounds of IDEA (Extended Abstract)*, technical report 93/1, Department of Electrical Engineering, ESAT–COSIC, Belgium, 1993.
19. Joan Daemen, René Govaerts, Joos Vandewalle, *Weak Keys for IDEA*, Advances in Cryptology, proceedings of CRYPTO '93, Lecture Notes in Computer Science 773, pp. 224–231, Springer-Verlag, 1994.
20. Joan Daemen, Lars R. Knudsen, Vincent Rijmen, *The Block Cipher Square*, proceedings of Fast Software Encryption 4, Lecture Notes in Computer Science 1267, pp. 149–165, Springer-Verlag, 1997.
21. Hüseyin Demirci, *Square-like Attacks on Reduced Rounds of IDEA*, proceedings of Selected Areas in Cryptography 2002, Lecture Notes in Computer Science 2595, pp. 147–159, Springer-Verlag, 2003.
22. Hüseyin Demirci, Ali A. Selçuk, Erkan Türe, *A New Meet-in-the-Middle Attack on the IDEA Block Cipher*, proceedings of Selected Areas in Cryptography 2003, Lecture Notes in Computer Science 3006, pp. 117–129, Springer-Verlag, 2004.
23. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting, *Improved Cryptanalysis of Rijndael*, proceeding of Fast Software Encryption 8, Lecture Notes in Computer Science 1978, pp. 213–230, Springer-Verlag, 2001.
24. Soichi Furuya, *Slide Attacks with a Known-Plaintext Cryptanalysis*, proceedings of Information and Communication Security 2001, Lecture Notes in Computer Science 2288, pp. 214–225, Springer-Verlag, 2002.
25. Andrew Granville, *Cycle lengths in a permutation are typically Poisson distributed*, available on-line at: *http://www.dms.umontreal.ca/~andrew/PDF/CycleLengths.pdf*
26. Philip Hawkes, *Differential-Linear Weak Keys Classes of IDEA*, Advances in Cryptology, proceedings if EUROCRYPT '98, Lecture Notes in Computer Science 1403, pp. 112–126, Springer-Verlag, 1998.
27. P. Hawkes, L. O'Connor, *On Applying Linear Cryptanalysis to IDEA*, Advances in Cryptology - Proceedings of ASIACRYPT'96, Lecture Notes in Computer Science 1163, pp. 105–115, Springer-Verlag, 1996.
28. Seokhie Hong, Jongsung Kim, Guil Kim, Sangjin Lee, Bart Preneel, *Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 368–383, Springer-Verlag 2005.
29. Tetsu Iwata, Tadayoshi Kohno, *New Security Proofs for the 3GPP Confidentiality and Integrity Algorithms*, proceedings of Fast Software Encryption 11, Lecture Notes in Computer Science 3017, pp. 427–445, Springer-Verlag, 2004.
30. Goce Jakimoski, Yvo Desmedt, *Related-Key Differential Cryptanalysis of 192-bit Key AES Variants*, proceedings of Selected Areas in Cryptography 2003, Lecture Notes in Computer Science 3006, pp. 208–221, Springer-Verlag, 2004.
31. Pascal Junod, *New Attacks Against Reduced-Round Versions of IDEA*, proceedings of Fast Software Encryption 12, Lecture Notes in Computer Science 3557, pp. 384–397, Springer-Verlag, 2005.
32. John Kelsey, Bruce Schneier, David Wagner, *Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Advances in Cryptology, proceedings of CRYPTO '96, Lecture Notes in Computer Science 1109, pp. 237–251, Springer-Verlag, 1996.

33. John Kelsey, Bruce Schneier, David Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*, proceedings of Information and Communication Security 1997, Lecture Notes in Computer Science 1334, pp. 233–246, Springer-Verlag, 1997.
34. Jongsung Kim, Seokhie Hong, Bart Preneel, *Related-Key Rectangle Attacks on Reduced AES-192 and AES-256*, accepted to Fast Software Encryption 14, private communications.
35. Jongsung Kim, Guil Kim, Seokhie Hong, Dowon Hong, *The Related-Key Rectangle Attack — Application to SHACAL-1*, proceedings of ACISP 2004, Lecture Notes in Computer Science 3108, pp. 123–136, Springer-Verlag, 2004.
36. Lars R. Knudsen, *Cryptanalysis of LOKI91*, proceedings of Auscrypt 1992, Lecture Notes in Computer Science 718, pp. 196–208, Springer-Verlag, 1993.
37. Xuejia Lai, James L. Massey, Sean Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology - Proceedings of EUROCRYPT'91, Lecture Notes in Computer Science 547, pp. 17–38, Springer-Verlag, 1992.
38. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 386–397, Springer-Verlag, 1994.
39. Willi Meier, *On the Security of the IDEA Block Cipher*, Advances in Cryptology, proceedings of EUROCRYPT '93, Lecture Notes in Computer Science 765, pp. 371–385, Springer-Verlag, 1994.
40. Jorge Nakahara Jr., Paulo S.L.M. Barreto, Bart Preneel, Joos Vandewalle, Hae Y. Kim, *SQUARE Attacks Against Reduced-Round PES and IDEA Block Ciphers*, IACR Cryptology ePrint Archive, Report 2001/068, 2001.
41. Jorge Nakahara Jr., Bart Preneel, Joos Vandewalle, *The Biryukov-Demirci Attack on Reduced-Round Versions of IDEA and MESH Ciphers*, proceedings of ACISP 2004, Lecture Notes in Computer Science 3108, pp. 98–109, Springer-Verlag, 2004.
42. NESSIE, *Performance of Optimized Implementations of the NESSIE Primitives*, NES/DOC/TEC/WP6/D21/a, available on-line at *http://www.nessie.eu.org/nessie*.
43. Havard Raddum, *Cryptanalysis of IDEA-X/2*, proceedings of Fast Software Encryption 10, Lecture Notes in Computer Science 2887, pp. 1–8, Springer-Verlag, 2003.
44. Gene Tsudik, Els Van Herreweghen, *On simple and secure key distribution*, Conference on Computer and Communications Security, Proceedings of the 1st ACM conference on Computer and communications security, pp. 49–57, ACM, 1993.
45. David Wagner, *The Boomerang Attack*, proceedings of Fast Software Encryption 6, Lecture Notes in Computer Science 1636, pp. 156–170, 1999.
46. David J. Wheeler, Roger M. Needham, *TEA, a Tiny Encryption Algorithm*, proceedings of Fast Software Encryption 2, Lecture Notes in Computer Science 1008, pp. 363–366, 1995.
47. ZDNet, New Xbox security cracked by Linux fans, available on-line at `http://news.zdnet.co.uk/software/developer/0,39020387,2123851,00.htm`, 2002.

A A Description of IDEA

IDEA is a 64-bit, 8.5-round block cipher with 128-bit keys proposed by Lai and Massey in 1991 [37].

Each round of IDEA (besides the last one) consists of two layers. Let the input of round i be denoted by four 16-bit words $(X_1^i, X_2^i, X_3^i, X_4^i)$. The first layer, denoted by KA, affects each word independently: The first and the fourth words are multiplied by subkey words (mod $2^{16}+1$) where 0 is replaced by 2^{16}, and the second and the third words are added with subkey words (mod 2^{16}). We denote the intermediate values after this half-round by $(Y_1^i, Y_2^i, Y_3^i, Y_4^i)$. Let Z_1^i, Z_2^i, Z_3^i, and Z_4^i be the four subkey words, then

$$Y_1^i = Z_1^i \odot X_1^i; \quad Y_2^i = Z_2^i \boxplus X_2^i; \quad Y_3^i = Z_3^i \boxplus X_3^i; \quad Y_4^i = Z_4^i \odot X_4^i,$$

where $\odot$ denotes multiplication modulo $2^{16}+1$ with 0 replaced by 2^{16}, and where $\boxplus$ denotes addition modulo 2^{16}.

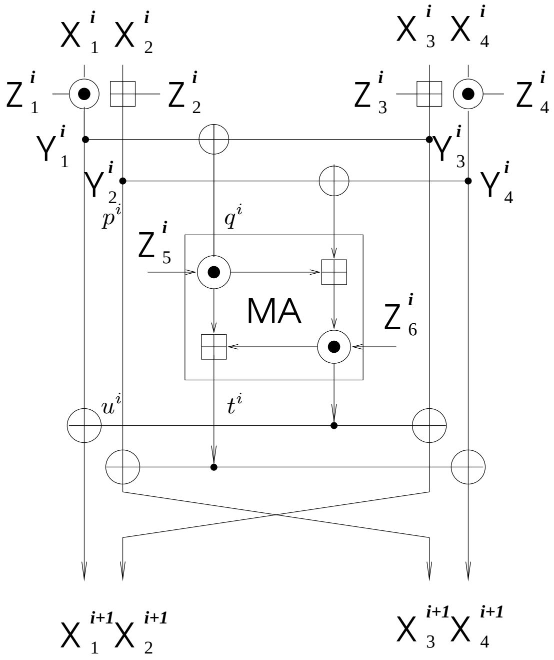


Fig. 3. One Round of IDEA

The second layer, denoted by MA, accepts two 16-bit words p^i and q^i computed as $(p^i, q^i) = (Y_1^i \oplus Y_3^i, Y_2^i \oplus Y_4^i)$. We denote the two output words of the MA transformation by (u^i, t^i). Denoting the subkey words that enter the MA function of round i by Z_5^i and Z_6^i,

$$t^i = (q^i \boxplus (p^i \odot Z_5^i)) \odot Z_6^i; \quad u^i = (p^i \odot Z_5^i) \boxplus t^i$$

The output of the i-th round is $(Y_1^i \oplus t^i, Y_3^i \oplus t^i, Y_2^i \oplus u^i, Y_4^i \oplus u^i)$. In the last round (round 9) the MA layer is removed. Thus, the ciphertext is $(Y_1^9||Y_2^9||Y_3^9||Y_4^9)$. The structure of a single round of IDEA is shown in Figure 3.

IDEA's key schedule expands the 128-bit key into $6 \cdot 8 + 4 = 52$ subkeys of 16 bits each using a very simple algorithm. The key is first used as the first eight subkeys. Then, the key is rotated by 25 bits to the left, and the outcome is used as the next eight subkeys. The rotation by 25 bits to the left is repeatedly used as many times as needed. All the subkeys for 8.5-round IDEA are listed in Table 2.

B A 4-Round Related-Key Truncated Differential of IDEA

Our attack exploits a 4-round related-key truncated differential. This related-key differential has probability 2^{-17} when a 16-bit condition on the plaintext is imposed, as we describe later.

The differential holds for rounds 1–4 of IDEA, and thus can be used as a building block in the unified related-key attack. The key difference is in bits 25 and 48 (i.e., the two related keys K^1 and K^2 satisfy $K^1 \oplus K^2 = \Delta K = e_{25,48}$).

The input difference of the differential is $\Delta_{IN} = (0, 8040_x, 0, 0)$, and the fourth input words of both plaintexts are required to be 1. Thus, after the first key addition layer, the difference

Round	Z_1^i	Z_2^i	Z_3^i	Z_4^i	Z_5^i	Z_6^i
$i = 1$	0–15	16–31	32–47	48–63	64–79	80–95
$i = 2$	96–111	112–127	25–40	41–56	57–72	73–88
$i = 3$	89–104	105–120	121–8	9–24	50–65	66–81
$i = 4$	82–97	98–113	114–1	2–17	18–33	34–49
$i = 5$	75–90	91–106	107–122	123–10	11–26	27–42
$i = 6$	43–58	59–74	100–115	116–3	4–19	20–35
$i = 7$	36–51	52–67	68–83	84–99	125–12	13–28
$i = 8$	29–44	45–60	61–76	77–92	93–108	109–124
$i = 9$	22–37	38–53	54–69	70–85		

Table 2. The Key Schedule Algorithm of IDEA

becomes $(0, 8000_x, 0, 8000_x)$ with probability 1/2. We note that the difference in the most significant bit of the fourth word is caused by the key difference and the fact that the fourth words of the plaintexts have the value of 1.

The input difference to the first MA layer is $(0, 0)$, and thus, the output difference of the first round is $(0, 0, 8000_x, 8000_x)$. In the second KA layer, the difference in the third word is cancelled by the key difference with probability 1. Under the assumption that the multiplication operation (under the two related subkeys) has a close to random behavior, the difference in the fourth word is cancelled as well with probability 2^{-16}. Hence, with probability 2^{-17} there is a zero difference after the second KA layer.

The zero difference state remains until the MA layer of the fourth round, where the key difference affects the two subkeys. Thus, the output difference of the related-key differential is $\Delta_{OUT} = (a, a, b, b)$ for some unknown a and b.

Note that if the differential is satisfied, then the difference before the MA layer of the fourth round is a zero difference. This property is used in the key deduction phase of the attack presented in Section 4.

C Another Key Recovery Attack on 4*r*-Round IDEA

In this section we propose a different attack algorithm on $4r$-round IDEA. The attack is different than the one in Section 4.2 in the way the chains are used to detect the related-key plaintext pairs. As before, we present the attack on 8-round IDEA, but it can be easily transformed to other $4r$-round variants of IDEA.

C.1 Attack Algorithm

1. **Data Generation:**
   (a) Pick a plaintext P_0^0 randomly, and ask for its encryption under the unknown key K and for the encryption of the plaintext $R_0^0 = P_0^0 \oplus \Delta_{IN}$ under $K' = K \oplus \Delta K$. Denote the corresponding ciphertexts by P_0^{22} and R_0^{22}, respectively. Then ask for the encryption of P_0^{22} under $K \lll 22$ and of R_0^{22} under $K' \lll 22$, and denote the corresponding ciphertexts by P_0^{44} and R_0^{44}, respectively. Continue the process until the keys are again K and K', and denote the plaintexts by P_1^0 and R_1^0, respectively. Repeat the process until 2^{20} pairs (P_i^0, R_j^0) such that $P_i^0 \oplus R_j^0 = \Delta_{IN}$ with the fourth word of P_i^0 and R_j^0 equal to 1 are encountered. Store each such set of indices (i, j) in a table $Table_{P,R}$. In case the chains end before enough such pairs are encountered, another P_0^0 and R_0^0 are chosen.
   (b) Repeat Steps 1.2 and 1.4 of the attack from Section 4.2 to obtain $Chain_Q$ and $Chain_S$.

2. **Generating Related-Key Plaintext Pairs:** Choose a pool of 2^{32} candidate related-key plaintext counterparts to R_0^0 denoted by $\{S_m^{75}\}_{m=0}^{2^{32}-1}$, such that $\forall m, m'$ the difference $S_m^{75} \oplus S_{m'}^{75}$ is in Δ_{OUT}. Note that the entire space of plaintexts is divided to 2^{32} disjoint pools of this form. For each pool perform the following:
   (a) For each $(i,j) \in Table_{P,R}$, and for all m, compute the encryption of S_m^{75} j positions further in the chain that starts from it, denoted by $S_{m,j}^{75}$. Store in a table the values of $S_{m,j}^{75}$, along with S_m^{75}.
   (b) Compute the set of 2^{32} possible Q_l^{75} such that $S_m^{75} \oplus Q_l^{75} \in \Delta_{OUT}$. (We note that the pool $\{Q_l^{75}\}_{l=0}^{2^{32}-1}$ is actually equal to the pool $\{S_m^{75}\}_{m=0}^{2^{32}-1}$, since Δ_{OUT} is closed under the XOR operation).
   (c) For every $(i,j) \in Table_{P,R}$ and for all l, compute the i'th places in the chain of the encryption of Q_l^{75}, denoted by $Q_{l,i}^{75}$.
   (d) For every $(i,j) \in Table_{P,R}$, and for each pair (Q_l^{75}, S_m^{75}), check how many times $Q_{l,i}^{75} \oplus S_{m,j}^{75} \in \Delta_{OUT}$ is satisfied. If this number is greater than 4, apply the key recovery algorithm using Q_l^{75} as the related-key plaintext counterpart of P_0^0 and S_m^{75} as the related-key plaintext counterpart of R_0^0.
3. If the attack fails for all the 2^{32} pools of $\{S_m^{75}\}$, repeat Step 1.1 and Step 2 for another choice of P_0^0 and R_0^0.

The attack identifies the correct related-key plaintext pairs by observing the fact that if Q_l^{75} and S_m^{75} are the related-key plaintext counterparts of P_0^0 and R_0^0, respectively, then out of the 2^{20} checked pairs, it is expected that 8 satisfy Δ_{OUT}. In case that (P_0^0, Q_l^{75}) and (R_0^0, S_m^{75}) are not related-key plaintext pairs, then only 2^{-12} of the $Q_{l,i}^{75} \oplus S_{m,j}^{75}$ values are expected to satisfy Δ_{OUT}.

C.2 Analysis of the Attack

We first note that during the attack, Steps 1 and 2 are expected to be executed 2^{17} times (until encountering a pair (P_0^0, R_0^0) that satisfies the related-key differential).

Step 1 consists mainly of searching the encryption chains of P_0^0 and R_0^0, which can be efficiently performed using hash tables. Thus, Step 1 requires about 2^{21} memory accesses for finding the pairs (i,j), and another 2^{20} memory accesses for storing them.

For each pool $\{S_m^{75}\}_{m=0}^{2^{32}-1}$, steps 2(a),2(b), and 2(c) consist of generating sets of 2^{20} values 2^{33} times (2^{32} for S_m^{75}'s and 2^{32} times for Q_l^{75}'s) . Assuming that these values are generated as in the attack of Section 4.2, this stage is mainly composed of memory accesses (2^{53} for each set of S_m^{75}'s). Step 2(d) is the more complex one. By correctly indexing the tables, it is possible to try all the possible pairs of the form $(Q_{l,i}^{75}, S_{m,j}^{75})$ requiring only 2^{52} memory accesses for a pool. This is done by storing for each j all the values $S_{m,j}^{75}$ (or more precisely the 32 bits composed of the XOR of the first two words and the XOR of the last two words of $S_{m,j}^{75}$ for all m's). Then to query a specific $Q_{l,i}^{75}$ it is sufficient to compute the XOR of the first and second words, and the XOR of the third and fourth words, and to check whether this value appears in the corresponding table. Thus, we conclude that Step 2 requires 2^{54} memory accesses for any pool of S_m^{75}'s. As there are 2^{32} such pools, the total time complexity of Step 2 is 2^{86} memory accesses for each choice of the pair (P_0^0, R_0^0).

Therefore, we conclude that this attack requires 2^{103} memory accesses, which is roughly the same as the attack in Section 4.2. The data complexity of the attack is about 2^{71} related-key chosen plaintexts and ciphertexts for obtaining $Chain_Q$ and $Chain_S$. Generating the chains of all the pairs of (P_0^0, R_0^0) required for the attack is expected to require about $2^{65.5}$ related-key adaptive chosen plaintexts and ciphertexts.