J. Cryptol. (2014) 27: 824–849
DOI: 10.1007/s00145-013-9154-9

Journal of
CRYPTOLOGY

A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony

Orr Dunkelman

Computer Science Department, University of Haifa, Haifa 31905, Israel
orrd@cs.haifa.ac.il
and
Faculty of Mathematics and Computer Science, Weizmann Institute of Science, P.O. Box 26, Rehovot 76100, Israel

Nathan Keller*

Faculty of Mathematics and Computer Science, Weizmann Institute of Science, P.O. Box 26, Rehovot 76100, Israel
nathan.keller@weizmann.ac.il
and
Department of Mathematics, Bar Ilan University, Ramat Gan 52900, Israel

Adi Shamir

Faculty of Mathematics and Computer Science, Weizmann Institute of Science, P.O. Box 26, Rehovot 76100, Israel
adi.shamir@weizmann.ac.il

Communicated by Kaisa Nyberg

Received 14 November 2011
Online publication 9 July 2013

Abstract. Over the last 20 years, the privacy of most GSM phone conversations was protected by the A5/1 and A5/2 stream ciphers, which were repeatedly shown to be cryptographically weak. They are being replaced now by the new A5/3 and A5/4 algorithms, which are based on the block cipher KASUMI. In this paper we describe a new type of attack called a *sandwich attack*, and use it to construct a simple related-key distinguisher for 7 of the 8 rounds of KASUMI with an amazingly high probability of 2^{-14}. By using this distinguisher and analyzing the single remaining round, we can derive the complete 128-bit key of the full KASUMI with a related-key attack which uses only 4 related keys, 2^{26} data, 2^{30} bytes of memory, and 2^{32} time. These completely practical complexities were experimentally verified by performing the attack in less than two hours on a single-core of a PC. Interestingly, neither our technique nor any other published attack can break the original MISTY block cipher (on which KASUMI is based) significantly faster than exhaustive search. Our results thus indicate that the modifications made by ETSI's SAGE group in moving from MISTY to KASUMI made it extremely weak when related-key attacks are allowed, but do not

* N. Keller was partially supported by the Koshland center for basic research.

imply anything about its resistance to single-key attacks. Consequently, there is no indication that the way KASUMI is implemented in GSM and 3G networks is practically vulnerable in any realistic attack model.

Key words. KASUMI, Sandwich attack, GSM/3G security, Related-key, Boomerang attack

1. Introduction

The privacy of GSM cellular telephony is protected by the A5 family of cryptosystems. The first two members of this family, the stream ciphers A5/1 (developed primarily for European markets) and A5/2 (developed primarily for export markets) were designed in the late 1980s in an opaque process and were kept secret until they were reverse engineered in 1999 from actual handsets [12]. Once published, it became clear that A5/2 provided almost no security, and A5/1 could be attacked with practical complexity by a variety of techniques (e.g., [2,3,10,13]). In particular, a team of cryptographers led by Karsten Nohl published in December 2009 a 2-TBytes rainbow table for A5/1, that makes it easy to derive the session key of any particular conversation with minimal delay and hardware support [1].

In response to these developments, the GSM Association decided to design a new block cipher with 128-bit keys called KASUMI [24], and to use it for both secrecy and authentication purposes, deploying newly developed modes of operation. This time, the process was significantly more open, and resulted in two ways to deploy KASUMI: A5/3 (using a simplified 64-bit key version of KASUMI) which is mandatory in all new handsets, and A5/4 (using the full 128-bit key version of KASUMI) which is optional and does not seem to be in use by any operator. In UMTS (3G) cellular networks, there are two possible encryption algorithms which are both mandatory on all handsets: UEA1 which is based on 128-bit KASUMI, and UEA2 which is based on 128-bit SNOW 3G. A5/3 and UEA1 are already implemented in a majority of the five billion available handsets, and thus KASUMI had become one of the most widely deployed cryptosystems in the world, and its security had become one of the most important practical issues in cryptography.

The KASUMI block cipher is based on the MISTY block cipher which was published at FSE 1997 by Matsui [20]. It has 64-bit blocks, 128-bit keys, and a complex recursive Feistel structure with 8 rounds, each one of which consists of 3 rounds, each one of which has 3 rounds of nonlinear SBox operations. MISTY withstood 15 years of cryptanalytic efforts, and only recently a first attack faster than exhaustive search on its full version has appeared, with a completely impractical complexity of 2^{125} [16]. However, the designers of A5/3 decided to make MISTY faster and more hardware-friendly by simplifying its key schedule and modifying some of its components. In [25], the designers provide a rational for each one of these changes, and in particular they analyze the resistance of KASUMI against related-key attacks [4] by stating that "removing all the FI functions in the key scheduling part makes the hardware smaller and/or reduces the key set-up time. We expect that related-key attacks do not work for this structure." The best attack found by the designers and external evaluators of KASUMI is described as follows: "There are chosen plaintext and/or related-key attacks against KASUMI reduced to 5 rounds. We believe that with further analysis it might be possible to extend some attacks to 6 rounds, but not to the full 8-round KASUMI."

The existence of better related-key attacks on the full KASUMI was already shown in [7]. The attack of [7] had a data complexity of $2^{54.6}$ and time complexity of $2^{76.1}$, which are impractical but better than exhaustive search. In this paper we develop a new attack, which requires only 4 related keys, 2^{26} data, 2^{30} bytes of memory, and 2^{32} time. Since these complexities are so low, we verified our attack experimentally, and our unoptimized implementation on a single core of an old PC recovered about 96 key bits in a few minutes, and the complete 128-bit key in less than two hours.[1] Careful analysis of our attack technique indicates that it cannot be applied against the original MISTY, since it exploits a sequence of coincidences and lucky strikes which were created when MISTY was changed to KASUMI by ETSI's SAGE task force working for the GSM Association. This calls into question the design of KASUMI, and especially its simplified key schedule.

In this paper, we develop a new type of attack which is an improved version of the boomerang attack introduced in [26]. We call it a "sandwich attack," since it uses a distinguisher which is divided into three parts: A thick slice ("bread") at the top, a thin slice ("meat") in the middle, and a thick slice ("bread") at the bottom. The top and bottom parts are assumed to have high probability differential characteristics, which can be combined into a quartet by the standard boomerang technique. However, in our case they are separated by an additional middle slice, which can significantly reduce the probability of the resulting boomerang structure. Nevertheless, as we show in this paper, careful analysis of the dependence between the top and bottom differentials allows us in some cases to combine the two properties above and below the middle slice with an enhanced probability. In particular, we show that in the case of KASUMI we can use top and bottom 3-round differential characteristics with an extremely high probability of 2^{-2} each, and combine them via a middle 1-round slice in such a way that the "cost in probability" of the combination is 2^{-6}, instead of the 2^{-32} we would expect from a naive analysis. This increases the probability of our 7-round distinguisher from 2^{-40} to 2^{-14}, and reduces significantly the data and the time complexities of the attack. Such a three-level structure was used in several previous attacks such as [8,9] (where it was called the "Feistel switch" or the "middle-round S-box trick"), but to the best of our knowledge it was always used in the past in simpler situations in which the transition probability through the middle layer (in at least one direction) was 1 due to the structural properties of a single Feistel round, or due to the particular construction of a given S-Box. Our sandwich attack is the first non-trivial application of such a structure, and the delicacy of the required probabilistic analysis is demonstrated by the fact that a tiny change in the key schedule of KASUMI or in the differentials (which both have no effect on the differential probabilities of the top and bottom layers) can change the probability of the combined distinguisher from the surprisingly high value of 2^{-14} to 0.

We note that after the sandwich technique was presented in the Crypto 2010 version of our paper, it was successfully applied to attack the MMB block cipher in [15]. We expect that other uses of this technique will be found in the future.

This paper is organized as follows: Section 2 describes the new sandwich attack, along with a chosen-plaintext variant which we call "rectangle-like sandwich attack,"

[1] Our implementation of the attack used the official reference implementation of KASUMI [25], which is not optimized for exhaustive search.

and discusses the transition between the top and bottom parts of the cipher through the middle slice of the sandwich. Section 3 describes the KASUMI block cipher. Section 4 describes our new 7-round distinguisher for KASUMI which has a probability of 2^{-14}, and demonstrates its extreme sensitivity to tiny structural modifications. In Sect. 5 we use the new distinguisher to develop a practical-time key recovery attack on the full KASUMI cryptosystem. Finally, Section 6 concludes the paper.

2. Sandwich Attacks

In this section we describe the technique used in our attacks on KASUMI. We start with a description of the basic (related-key) boomerang attack, and then describe a new framework, which we call a (related-key) *sandwich attack*, that exploits the dependence between the underlying differentials to obtain a more accurate estimation of the probability of the distinguisher. Finally, we describe the chosen plaintext variant of the attack, which we call (related-key) *rectangle-like sandwich attack*. We note that the idea of using dependence between the differentials in order to improve the boomerang distinguisher was implicitly proposed by Wagner [26], and was also used in some simple scenarios in [8,9]. Therefore, our framework can be considered as a formal treatment and generalization of the ideas proposed in [8,9,26].

2.1. *The Basic Related-Key Boomerang Attack*

The related-key boomerang attack was introduced by Kim et al. [14,18], and independently by Biham et al. [6], as a transformation of the boomerang attack [26] to the related-key differential settings [17]. In this attack, the cipher is treated as a cascade of two sub-ciphers $E = E_1 \circ E_0$, and related-key differentials of E_0 and E_1 are combined into an adaptive chosen plaintext and ciphertext distinguisher for E.

Let us assume that there exists a related-key differential $\alpha \rightarrow \beta$ for E_0 under key difference ΔK_{ab} with probability p (i.e., $\mathbf{Pr}[E_{0(K)}(P) \oplus E_{0(K \oplus K_{ab})}(P \oplus \alpha) = \beta] = p$, where $E_{0(K)}$ denotes encryption through E_0 under the key K and the probability is taken over all possible plaintexts and keys). Similarly, we assume that there exists a related-key differential $\gamma \rightarrow \delta$ for E_1 under key difference ΔK_{ac} with probability q. The related-key boomerang distinguisher requires encryption/decryption under the secret key K_a, and under the related keys $K_b = K_a \oplus \Delta K_{ab}$, $K_c = K_a \oplus \Delta K_{ac}$, and $K_d = K_c \oplus \Delta K_{ab} = K_b \oplus \Delta K_{ac}$.

The attack is based on the following process:

1. Pick a random plaintext P_a, and let $P_b = P_a \oplus \alpha$.
2. Ask for the ciphertexts $C_a = E_{K_a}(P_a)$ and $C_b = E_{K_b}(P_b)$. Denote $C_c = C_a \oplus \delta$ and $C_d = C_b \oplus \delta$.
3. Ask for the plaintexts $P_c = E^{-1}_{K_c}(C_c)$ and $P_d = E^{-1}_{K_d}(C_d)$.
4. Check whether $P_c \oplus P_d = \alpha$.

The probability that the pair (P_a, P_b) is a right pair with respect to the first differential (i.e., the probability that the intermediate difference after E_0 equals β, as predicted by the differential) is p. Assuming independence, the probability that both pairs (C_a, C_c)

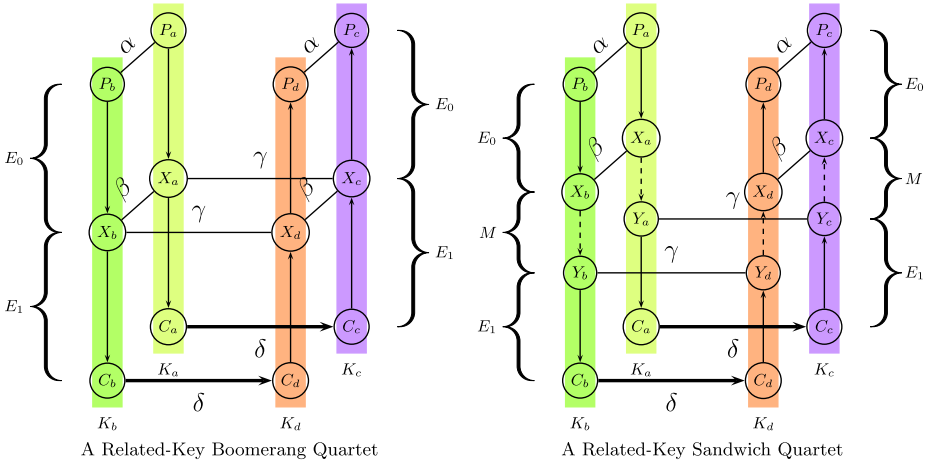

Fig. 1. Related-key boomerang and sandwich quartets.

and (C_b, C_d) are right pairs with respect to the second differential is q^2. If all these are right pairs, then we have

$$(X_a \oplus X_b = \beta) \wedge (X_a \oplus X_c = \gamma) \wedge (X_b \oplus X_d = \gamma),$$

where X_i is the intermediate encryption value of P_i. Thus,

$$X_c \oplus X_d = (X_c \oplus X_a) \oplus (X_a \oplus X_b) \oplus (X_b \oplus X_d) = \beta \oplus \gamma \oplus \gamma = \beta$$

(see left side of Fig. 1). This, in turn, implies that with probability p, $P_c \oplus P_d = \alpha$. Hence, the total probability of this quartet of plaintexts and ciphertexts to satisfy the condition $P_c \oplus P_d = \alpha$ is at least $(pq)^2$. For a random permutation the probability that the last condition is satisfied is 2^{-n}, where n is the block size. Therefore, if $pq \gg 2^{-n/2}$, it is possible to distinguish E from a random permutation given $O((pq)^{-2})$ adaptively chosen plaintexts and ciphertexts. The algorithm of the distinguisher is as follows:

1. Choose M plaintexts at random, and initialize a counter C to zero. For each plaintext P_a, perform the following:
   (a) Ask for the ciphertexts $C_a = E_{K_a}(P_a)$ and $C_b = E_{K_b}(P_b)$ where $P_b = P_a \oplus \alpha$.
   (b) Ask for the plaintexts $P_c = E_{K_c}^{-1}(C_c)$ and $P_d = E_{K_d}^{-1}(C_d)$ where $C_c = C_a \oplus \delta$ and $C_d = C_b \oplus \delta$.
   (c) If $P_c \oplus P_d = \alpha$, increment the counter C by 1.
2. If $C > \mathit{Threshold}$, output "E." Otherwise, output "Random Permutation."

The distinguisher can be improved by considering multiple differentials of the form $\alpha \to \beta'$ and $\gamma' \to \delta$ (for the same α and δ). We omit this improvement here since it is not used in our attack on KASUMI, and refer the reader to [6]. For a rigorous treatment of the related-key boomerang attack, including a discussion of the independence assumptions the attack relies upon, we refer the interested reader to [19,21].[2]

[2] In [21] it was shown that the independence assumptions underlying the attack may fail in various cases, and hence it is desirable to check the validity of the assumptions experimentally in each specific case. In the

The way to transform a related-key boomerang distinguisher into a key-recovery attack is rather standard, and thus we do not present it here and rely on the detailed description of such a transformation in our attack on KASUMI presented in Sect. 5.

2.2. *The Related-Key Sandwich Attack*

In this framework we consider the cipher as a cascade of three sub-ciphers: $E = E_1 \circ M \circ E_0$. Our assumptions are the same as in the basic boomerang attack: We assume that there exists a related-key differential $\alpha \to \beta$ for E_0 under key difference ΔK_{ab} with probability p, and a related-key differential $\gamma \to \delta$ for E_1 under key difference ΔK_{ac} with probability q. The attack algorithm is also exactly the same as in the basic attack (ignoring the middle sub-cipher M). However, the analysis is more delicate and requires great care in analyzing the dependence between the various distributions.

The main idea behind the sandwich attack is the transition in the middle. In the basic boomerang attack, if the pair (P_a, P_b) is a right pair with respect to the first differential, and both pairs (C_a, C_c) and (C_b, C_d) are right pairs with respect to the second differential, then we have

$$(X_a \oplus X_b = \beta) \wedge (X_a \oplus X_c = \gamma) \wedge (X_b \oplus X_d = \gamma), \tag{1}$$

where X_i is the intermediate encryption value of P_i, and thus

$$X_c \oplus X_d = (X_c \oplus X_a) \oplus (X_a \oplus X_b) \oplus (X_b \oplus X_d) = \beta \oplus \gamma \oplus \gamma = \beta, \tag{2}$$

resulting in $P_c \oplus P_d = \alpha$ with probability p (see left side of Fig. 1).

In the new sandwich framework, instead of condition (1), we get

$$(X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma), \tag{3}$$

where X_i is the partial encryption of P_i under E_0 (and the respective key) and Y_i is the partial decryption of C_i under E_1 (see right side of Fig. 1). Therefore, the probability of the three-layer related-key boomerang distinguisher is $p^2 q^2 r$, where

$$r = \mathbf{Pr}\big[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\big]. \tag{4}$$

Without further assumptions on M, r is expected to be very low (close to 2^{-n} for an n-bit block), and thus the distinguisher is expected to fail. However, as observed in [8,9,26], in some cases the differentials in E_0 and E_1 can be chosen such that the probability penalty r in going through M (in at least one direction) is 1, which is much higher than expected.

An example of this phenomenon, introduced in [26] and described in [9] under the name "Feistel switch," is the following. Let E be a Feistel cipher, decomposed as $E = E_1 \circ M \circ E_0$, where M consists of one Feistel round (see Fig. 2). Assume that the

case of KASUMI considered in this paper, we have verified both the distinguisher and the attack experimentally, and the probabilities match the theoretical prediction with high precision (as described in Sect. 4.2).

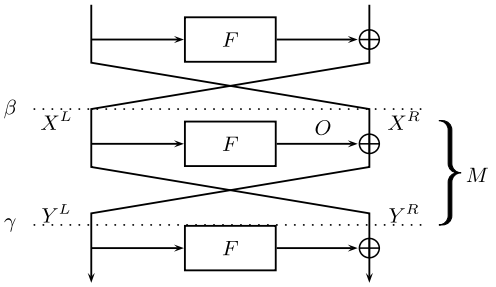

Fig. 2. A Feistel construction. M is the second round.

differentials $\alpha \to \beta$ (for E_0) and $\gamma \to \delta$ (for E_1) have no key difference (i.e., $\Delta K_{ab} = \Delta K_{ac} = 0$), and satisfy $\beta^L = \gamma^R$ (i.e., the left half of β which is the difference in the state X^L equals the right half of γ which is the difference in the state Y^R). We would like to compute the value of r.

Assume that condition (3) holds. In this case, as by the Feistel construction, $Y_i^R = X_i^L$ for all i, we have

$$X_a^L \oplus X_b^L = \beta^L = \gamma^R = X_a^L \oplus X_c^L = X_b^L \oplus X_d^L, \tag{5}$$

and thus,

$$\left(X_a^L = X_d^L\right) \quad \text{and} \quad \left(X_b^L = X_c^L\right). \tag{6}$$

Therefore, the output values of the F-function in the Feistel round represented by M, denoted in Fig. 2 by (O_a, O_b, O_c, O_d), satisfy

$$(O_a = O_d) \quad \text{and} \quad (O_b = O_c).$$

Since by the Feistel construction, $X_i^R = Y_i^L \oplus O_i$ and by condition (3), $Y_a \oplus Y_b \oplus Y_c \oplus Y_d = 0$, it follows that

$$X_a \oplus X_b \oplus X_c \oplus X_d = 0,$$

which by condition (3) implies $X_c \oplus X_d = \beta$. Thus, in this case we get that

$$r = \mathbf{Pr}\left[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\right] = 1,$$

independently of the choice of the F-function used.

Other examples of the same phenomenon are considered in [8] (under the name "middle-round S-box trick"), and in [9] (under the names "ladder switch" and "S-box switch").

Our attack on KASUMI is the first non-trivial example of this phenomenon in which a careful analysis shows that r is smaller than 1, but much larger than its expected value under the standard independence assumptions. In our attack, the cipher E (7-round KASUMI) is a Feistel construction, M consists of a single round, and $\beta^L = \gamma^R$. However, the argument presented above cannot be applied directly since there is a non-zero key difference in M, and thus a zero input difference to the F-function

does not imply a zero output difference. Instead, we analyze the F-function thoroughly and show that in this case, $r = 2^{-6}$ (instead of 2^{-32}, which is the expected value for a random Feistel round in a 64-bit block cipher).

Remark 1. We note that our treatment of the sandwich distinguisher allows us to specify the precise independence assumptions we rely upon. Since r is defined as a conditional probability, the only independence assumptions we use are between the differentials of E_0 and E_1, and thus the formula p^2q^2r relies on exactly the same assumptions as the ordinary boomerang attack. In [8,9,26], this situation was treated as a "trick" allowing to increase the probability of the distinguisher, or in other words, as a failure of the formula p^2q^2 in favor of the adversary. This approach is problematic since once we claim that the entire formula does not hold due to dependencies, we cannot rely on independence assumptions in other places where such dependencies could be found.

2.3. *The Rectangle-Like Sandwich Attack*

The transformation of the (related-key) boomerang distinguisher into a chosen plaintext rectangle attack relies on standard birthday-paradox arguments. The division into sub-ciphers and the assumptions are the same as in the (related-key) boomerang distinguisher. The key idea behind the transformation is to encrypt many plaintext pairs with input difference α, and to look for quartets that happen to conform to the requirements of the boomerang process. In other words, the adversary considers quartets of plaintexts of the form $((P_a, P_b = P_a \oplus \alpha), (P_c, P_d = P_c \oplus \alpha))$ encrypted under the related keys K_a, K_b, K_c, and K_d, respectively, and a quartet is called a "right quartet" if the following conditions are satisfied:

1. $E_{0(K_a)}(P_a) \oplus E_{0(K_b)}(P_b) = \beta = E_{0(K_c)}(P_c) \oplus E_{0(K_d)}(P_d)$ (i.e., $X_a \oplus X_b = \beta = X_c \oplus X_d$).
2. $E_{0(K_a)}(P_a) \oplus E_{0(K_c)}(P_c) = X_a \oplus X_c = \gamma$ (which leads to $E_{0(K_b)}(P_b) \oplus E_{0(K_d)}(P_d) = X_b \oplus X_d = \gamma$ if this condition holds along with the previous one).
3. $C_a \oplus C_c = \delta = C_b \oplus C_d$.

The probability of a quartet to be a right quartet is a lower bound on the probability of the event

$$C_a \oplus C_c = \delta = C_b \oplus C_d. \tag{7}$$

The usual assumption is that each of the above conditions is independent of the rest, and hence the probability that a given quartet $((P_a, P_b), (P_c, P_d))$ is a right quartet is $p^2 \cdot 2^{-n} \cdot q^2$. Since for a random permutation, the probability of condition (7) is 2^{-2n}, the rectangle process can be used to distinguish E from a random permutation if $pq \gg 2^{-n/2}$ (the same condition as in the standard boomerang distinguisher).

However, the data complexity of the distinguisher is $O(2^{n/2}(pq)^{-1})$, which is much higher than the complexity of the boomerang distinguisher. The higher data complexity follows from the fact that the event $E_{0(K_a)}(P_a) \oplus E_{0(K_c)}(P_c) = \gamma$ occurs with a "random" probability of 2^{-n} (in fact, this is the birthday-paradox argument behind the construction). The identification of right quartets is also more complicated than in the boomerang case, as instead of checking a condition on pairs, the adversary has to go

over all the possible quartets. At the same time, the chosen plaintext nature allows using stronger key recovery techniques. An optimized method of finding the right rectangle quartets is presented in [5].

The transformation of the (related-key) sandwich framework into the (related-key) rectangle-like sandwich framework is performed similarly. The way in which the distinguisher is deployed remains the same, and the probability of a quartet to be a right quartet is $p^2 \cdot 2^{-n} \cdot r' \cdot q^2$, where

$$r' = \mathbf{Pr}\big[(Y_b \oplus Y_d = \gamma) \mid (X_a \oplus X_b = \beta) \wedge (X_c \oplus X_d = \beta) \wedge (Y_a \oplus Y_c = \gamma)\big]. \quad (8)$$

It follows from symmetry arguments that in the case where E is a Feistel cipher, M consists of a single round, and $\beta^L = \gamma^R$, we have $r' = r$ (even if there is a non-zero key difference in M). Thus, in our attack on KASUMI we are able to use the computation of r in the sandwich framework to find also the probability of the corresponding related-key rectangle-like sandwich distinguisher.

3. The KASUMI Block Cipher

KASUMI [24] is a 64-bit block cipher with 128-bit keys. It has a recursive Feistel structure, following its ancestor MISTY. The cipher has eight Feistel rounds, where each round is composed of two functions: the *FO* function which is in itself a 3-round 32-bit Feistel construction, and the *FL* function that mixes a 32-bit subkey with the data in a linear way. The order of the two functions depends on the round number: in the even rounds the *FO* function is applied first, and in the odd rounds the *FL* function is applied first.

The *FO* function also has a recursive structure: its F-function, called *FI*, is a four-round Feistel construction. The *FI* function uses two nonlinear S-boxes $S7$ and $S9$ (where $S7$ is a 7-bit to 7-bit permutation and $S9$ is a 9-bit to 9-bit permutation), and accepts an additional 16-bit subkey, which is mixed with the data. In total, a 96-bit subkey enters *FO* in each round—48 subkey bits are used in the *FI* functions and 48 subkey bits are used in the key mixing stages.

The *FL* function accepts a 32-bit input and two 16-bit subkey words. One subkey word affects the data using the OR operation, while the second one affects the data using the AND operation. We outline the structure of KASUMI and its components in Fig. 3.

The key schedule of KASUMI is much simpler than the original key schedule of MISTY, and the subkeys are linearly derived from the key. The 128-bit key K is divided into eight 16-bit words: $K_1, K_2, \ldots, K_8$. Each K_i is used to compute $K'_i = K_i \oplus C_i$, where the C_i's are fixed constants (we omit these from the paper, and refer the intrigued reader to [24]). In each round, eight words are used as the round subkey (up to some in-word rotations). Hence, each 128-bit round subkey is a linearly modified version of the secret key. We summarize the details of the key schedule of KASUMI in Table 1.

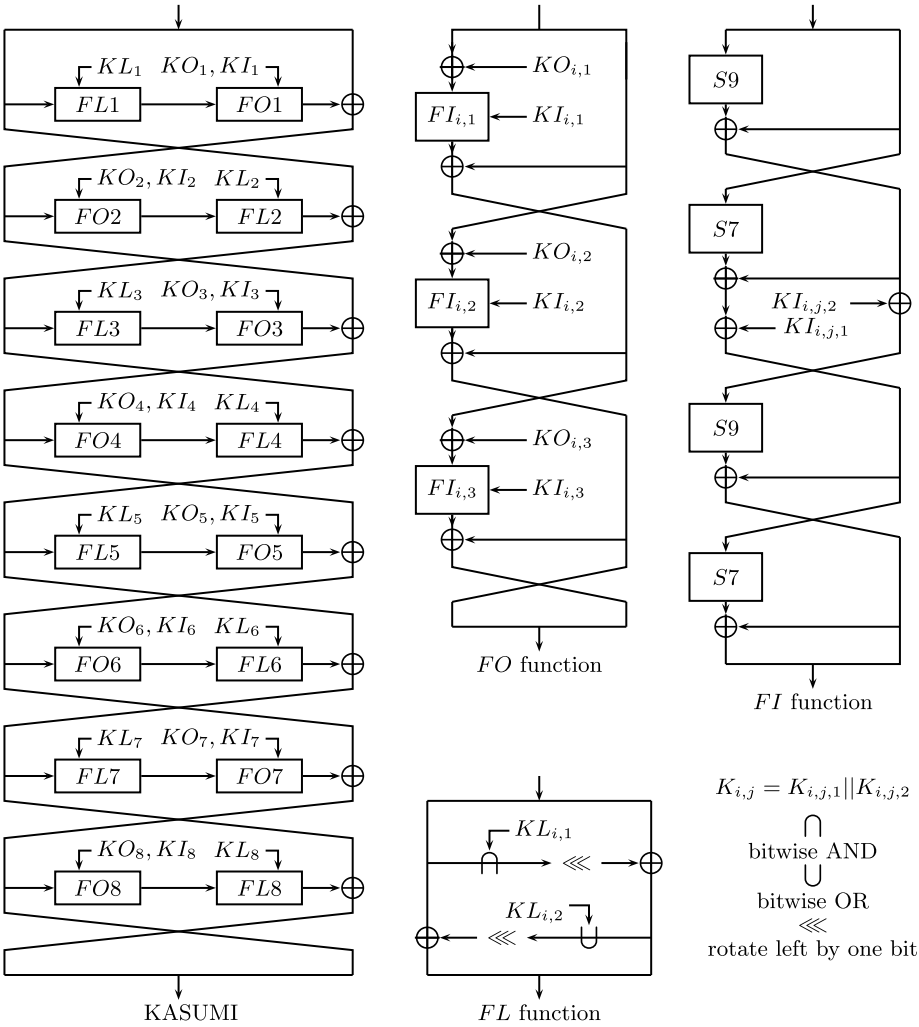

Fig. 3. Outline of KASUMI.

4. A Related-Key Sandwich Distinguisher for 7-Round KASUMI

4.1. *The New Distinguisher*

In our distinguisher, we treat rounds 1–7 of KASUMI as a cascade $E = E_1 \circ M \circ E_0$, where E_0 consists of rounds 1–3, M consists of round 4, and E_1 consists of rounds 5–7. The related-key differential we use for E_0 is a slight modification of the differential characteristic presented in [11], in which

$$\alpha = (0_x, 0010\ 0000_x) \rightarrow (0010\ 0000_x, 0_x) = \beta.$$

Table 1. KASUMI's key schedule algorithm.

Round	$KL_{i,1}$	$KL_{i,2}$	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$
1	$K_1 \lll 1$	K'_3	$K_2 \lll 5$	$K_6 \lll 8$	$K_7 \lll 13$	K'_5	K'_4	K'_8
2	$K_2 \lll 1$	K'_4	$K_3 \lll 5$	$K_7 \lll 8$	$K_8 \lll 13$	K'_6	K'_5	K'_1
3	$K_3 \lll 1$	K'_5	$K_4 \lll 5$	$K_8 \lll 8$	$K_1 \lll 13$	K'_7	K'_6	K'_2
4	$K_4 \lll 1$	K'_6	$K_5 \lll 5$	$K_1 \lll 8$	$K_2 \lll 13$	K'_8	K'_7	K'_3
5	$K_5 \lll 1$	K'_7	$K_6 \lll 5$	$K_2 \lll 8$	$K_3 \lll 13$	K'_1	K'_8	K'_4
6	$K_6 \lll 1$	K'_8	$K_7 \lll 5$	$K_3 \lll 8$	$K_4 \lll 13$	K'_2	K'_1	K'_5
7	$K_7 \lll 1$	K'_1	$K_8 \lll 5$	$K_4 \lll 8$	$K_5 \lll 13$	K'_3	K'_2	K'_6
8	$K_8 \lll 1$	K'_2	$K_1 \lll 5$	$K_5 \lll 8$	$K_6 \lll 13$	K'_4	K'_3	K'_7

$(X \lll i)$—X rotated to the left by i bits.

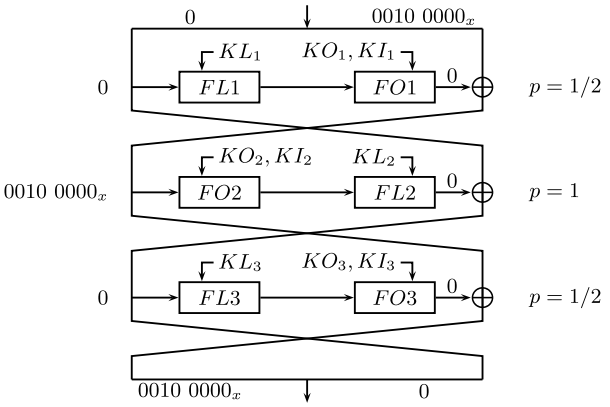

Fig. 4. 3-Round related-key differential characteristic of KASUMI.

The corresponding key difference is $\Delta K_{ab} = \Delta K_{cd} = (0, 0, 8000_x, 0, 0, 0, 0, 0)$, i.e., only the third key word has a single bit difference $\Delta K_3 = 8000_x$. This related-key differential is depicted in Fig. 4. The related-key differential we use for E_1 is the same differential shifted by four rounds, in which the data differences are

$$\gamma = (0_x, 0010\ 0000_x) \rightarrow (0010\ 0000_x, 0_x) = \delta,$$

and the key difference is $\Delta K_{ac} = \Delta K_{bd} = (0, 0, 0, 0, 0, 0, 8000_x, 0)$ (to handle the different subkeys used in these rounds).

As shown in [11], the probability of each one of these 3-round differential characteristics is $1/4$. In order to find the probability of the related-key sandwich distinguisher, we need to compute the probability

$$\mathbf{Pr}\big[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\big], \qquad (9)$$

where (X_a, X_b, X_c, X_d) and (Y_a, Y_b, Y_c, Y_d) are the intermediate values before and after the middle slice of the sandwich during the encryption/decryption of the quartet

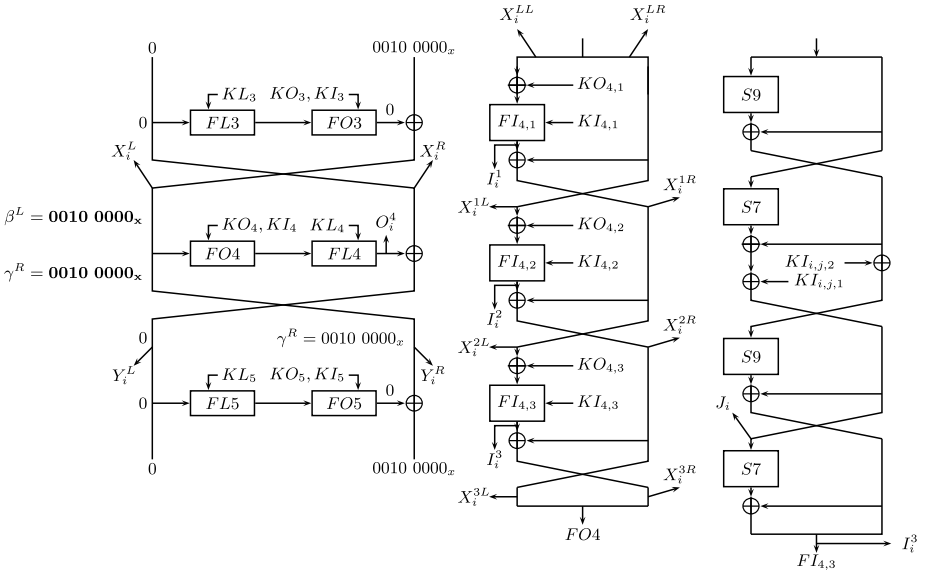

Fig. 5. Rounds 3–5 of the sandwich distinguisher and the notations used in the attack description.

(P_a, P_b, P_c, P_d) (see the right side of Fig. 1). This computation, which is a bit complex, spans the rest of this subsection.

Consider a quartet (P_a, P_b, P_c, P_d) for which the condition

$$(X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma) \tag{10}$$

is satisfied. Note that for our differentials, we have $\beta^L = \gamma^R$, as illustrated in Fig. 5. Hence, we can apply the argument of Sect. 2, since M is a single Feistel round. In particular, we obtain

$$\left(X_a^L = X_d^L\right) \wedge \left(X_b^L = X_c^L\right), \tag{11}$$

where X_i^L denotes the left half of X_i, which enters the function $FO4$ (see left part of Fig. 5). Moreover, as the right half of β^L and γ^R is zero, we have

$$X_a^{LR} = X_b^{LR} = X_c^{LR} = X_d^{LR}, \tag{12}$$

where X_i^{LR} denotes the right half (i.e., the 16 rightmost bits) of X_i^L (see central part of Fig. 5).

The following transitions are illustrated in Fig. 6 and the notations we use in their description are shown in Fig. 5. The function $FO4$ is a 3-round Feistel construction whose 32-bit values after round j are denoted by $(X_a^j, X_b^j, X_c^j, X_d^j)$. The functions $FI_{4,1}$, $FI_{4,2}$, and $FI_{4,3}$ are 4-round Feistel constructions, and the 16-bit outputs of $FI_{4,j}$ are denoted by $(I_a^j, I_b^j, I_c^j, I_d^j)$. Note that the key differences ΔK_{ab} and ΔK_{ac} affect in round 4 the subkeys $KI_{4,3}$ and $KI_{4,2}$, respectively, and in particular, there is no key difference in the first round of $FO4$. As a result, Eq. (11) implies that

$$\left(X_a^1 = X_d^1\right) \wedge \left(X_b^1 = X_c^1\right). \tag{13}$$

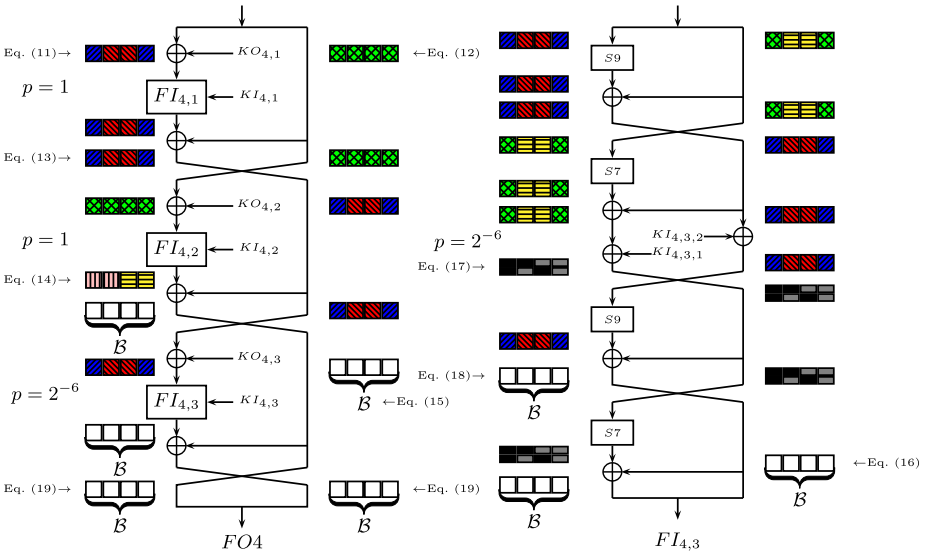

Fig. 6. The development of differences in FO_4 and in $FI_{4,3}$.

Furthermore, there is no key difference in the pairs corresponding to (P_a, P_b) and (P_c, P_d) in the second round of $FO4$, and thus Eq. (12) implies that:

$$\left(I_a^2 = I_b^2\right) \wedge \left(I_c^2 = I_d^2\right). \tag{14}$$

Combining Eqs. (13) and (14), we get the following relation in the right half of the intermediate values after round 3 of $FO4$:

$$X_a^{3R} \oplus X_b^{3R} \oplus X_c^{3R} \oplus X_d^{3R} = 0. \tag{15}$$

In the F-function of round 3 of $FO4$ we consider the pairs corresponding to (P_a, P_d) and (P_b, P_c). Since the key difference in these pairs (which equals to $K_{ab} \oplus K_{ac}$) affects only the subkey $KI_{4,3,1}$, Eq. (13) suggests that

$$I_a^{3R} \oplus I_b^{3R} \oplus I_c^{3R} \oplus I_d^{3R} = 0 \tag{16}$$

in the 9 bits which composes the right hand side of the output. In the left hand side of the output, the XOR of the four values is not necessarily equal to zero, due to the subkey difference that affects the inputs to the second $S7$ in $FI_{4,3}$. However, if these 7-bit inputs, denoted by (J_a, J_b, J_c, J_d), satisfy one of the conditions,

$$\left((J_a = J_b) \wedge (J_c = J_d)\right) \quad \text{or} \quad \left((J_a = J_c) \wedge (J_b = J_d)\right), \tag{17}$$

then Eq. (16) implies

$$I_a^{3L} \oplus I_b^{3L} \oplus I_c^{3L} \oplus I_d^{3L} = 0. \tag{18}$$

Since we have $J_a \oplus J_d = J_b \oplus J_c$ (both are equal to the subkey difference in $KI_{4,3,1}$), each one of the two conditions in Eq. (17) is expected to hold[3] with probability 2^{-7}. Therefore, combining Eqs. (15), (16), and (18) we get that the condition

$$X_a^3 \oplus X_b^3 \oplus X_c^3 \oplus X_d^3 = 0 \tag{19}$$

holds with probability 2^{-6}.

Finally, since the *FL* function is linear for a given key and there is no key difference in *FL4*, we can conclude that whenever Eq. (19) holds, the outputs of the F-function in round 4 (denoted in Fig. 5 by $(O_a^4, O_b^4, O_c^4, O_d^4)$) satisfy

$$O_a^4 \oplus O_b^4 \oplus O_c^4 \oplus O_d^4 = 0. \tag{20}$$

Since by condition (10),

$$Y_a^L \oplus Y_b^L \oplus Y_c^L \oplus Y_d^L = 0,$$

it follows that

$$X_a^R \oplus X_b^R \oplus X_c^R \oplus X_d^R = 0 \tag{21}$$

also holds with probability 2^{-6}. Combining this with Eq. (11) yields

$$\mathbf{Pr}\big[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\big] = 2^{-6}. \tag{22}$$

Therefore, the overall probability of the related-key sandwich distinguisher is

$$(1/4)^2 \cdot (1/4)^2 \cdot 2^{-6} = 2^{-14}, \tag{23}$$

which is much higher than the probability of $(1/4)^2 \cdot (1/4)^2 \cdot 2^{-32} = 2^{-40}$ which is expected by the naive analysis of the sandwich structure.

4.2. *Experimental Verification*

To verify the properties of the new distinguisher, we used the official code available as an appendix in [24]. The verification experiment was set up as follows: In each test we randomly chose a key quartet satisfying the required key differences. We then generated 2^{16} quartets by following the boomerang procedure described above. We utilized a slight improvement of the first differential suggested in [11] that increases its probability in the encryption direction by a factor of 2 by fixing the value of two plaintext bits. Hence, the number of right quartets in each test was expected to follow a Poisson distribution with a mean value of $2^{16} \cdot 2^{-14} \cdot 2 = 8$. We repeated the test 100,000 times, and obtained a distribution which is extremely close to the expected distribution. The full results are summarized in Table 2.

[3] This estimate is based on a randomness assumption that could be inaccurate in our case due to dependence between the differential characteristics. However, our experiments verify that this probability is indeed as expected.

Table 2. The number of right quartets in 100,000 experiments.

Right quartets	0	1	2	3	4	5	6	7	8
Theory (*Poi*(8))	34	268	1,073	2,863	5,725	9,160	12,214	13,959	13,959
Experiment	32	259	1,094	2,861	5,773	9,166	12,407	13,960	13,956

Right quartets	9	10	11	12	13	14	15	16	17
Theory (*Poi*(8))	12,408	9,926	7,219	4,813	2,962	1,692	903	451	212
Experiment	12,230	9,839	7,218	4,804	3,023	1,672	859	472	219

Right quartets	18	19	20	21	22	23	24	25
Theory (*Poi*(8))	94	40	16	6	2	0.8	0.26	0.082
Experiment	89	39	13	12	2	0	0	1

4.3. *A Tale of Two Sandwiches*

In this subsection we present two examples which demonstrate the extremely delicate nature of the probability estimations used in the sandwich attack, and the "lucky strikes" which made our attack on KASUMI possible. These two examples, along with a detailed analysis of various related-key boomerang distinguishers of a similar nature presented in [19], illustrate the thorough analysis of the structure of M which must be performed in each specific case in order to compute the probability r analytically. Another possibility is to give up the rigorous theoretical analysis and sample the probability r experimentally instead.

In the first example we present, we make a tiny change in the key schedule of KASUMI, which does not seem to have any effect on the differential probabilities of any one of its three sub-ciphers. However, for this example, the probability of the distinguisher is zero! In the second example, we use the original KASUMI key schedule, and slightly alter the differentials, such that the differential probabilities in the top and bottom sub-ciphers are not changed. As in the first example, it turns out that the probability of the distinguisher becomes zero.

4.3.1. *A Slight Change in the KASUMI Key Schedule*

The only change we make in KASUMI is the order of the subkeys. We take the original key schedule of KASUMI, and swap the roles of $KI_{i,1}$ and $KI_{i,3}$. Namely, the word used in KASUMI as $KI_{i,1}$ is used in this variant as $KI_{i,3}$ and vice versa. For example, in our variant $KI_{1,3} = K'_5$, $KI_{2,1} = K'_1$, and $KI_{3,3} = K'_7$.

Since our change affects only the subkeys used in $KI_{i,1}$ and $KI_{i,3}$ in each round, the differentials used in our distinguisher on KASUMI remain exactly the same for the new variant (with the same input/output differences, the same key differences and the same probabilities). However, we claim that in this case,

$$r = \mathbf{Pr}\big[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\big] = 0, \quad (24)$$

and thus the probability of the distinguisher is zero. In all the computations below, the notations are the same as in the original distinguisher above. The impossible transition is depicted in Fig. 7.

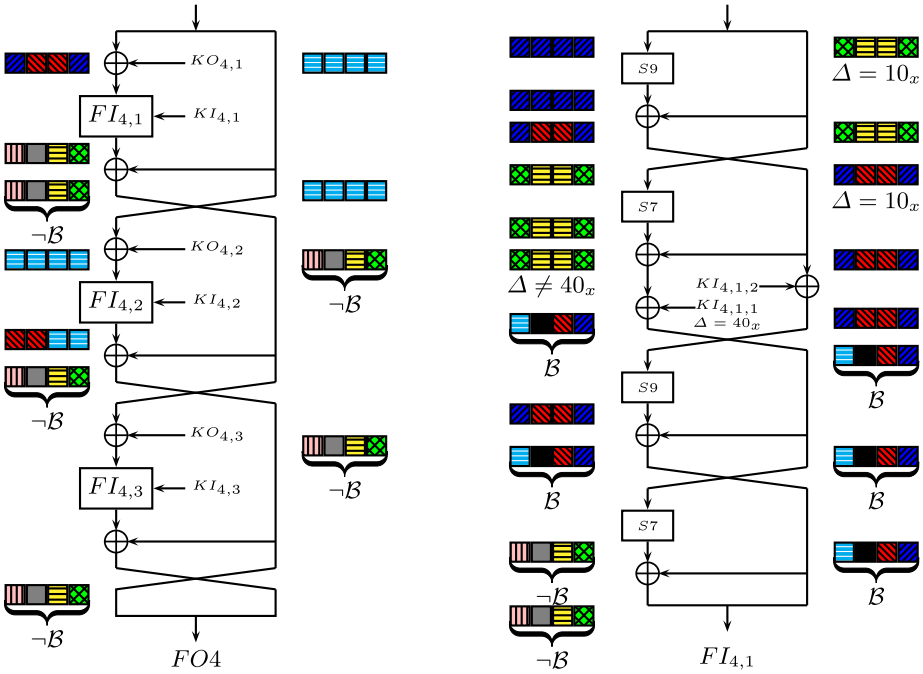

Fig. 7. The development of differences in FO_4 and in $FI_{4,1}$ in the modified KASUMI.

Since the differentials are the same as in the original distinguisher, we have

$$\left(X_a^L = X_d^L\right) \wedge \left(X_b^L = X_c^L\right) \tag{25}$$

and

$$X_a^{LR} = X_b^{LR} = X_c^{LR} = X_d^{LR}. \tag{26}$$

Also, since the second round of $FO4$ is unchanged, we have

$$\left(I_a^2 = I_b^2\right) \wedge \left(I_c^2 = I_d^2\right). \tag{27}$$

Therefore,

$$X_a^{3L} \oplus X_b^{3L} \oplus X_c^{3L} \oplus X_d^{3L} = I_a^1 \oplus I_b^1 \oplus I_c^1 \oplus I_d^1. \tag{28}$$

In the first round of $FO4$ we have a difference between the modified variant and the original KASUMI, as in the modified variant there is a subkey difference in the pairs corresponding to (P_a, P_b) and to (P_c, P_d), in the MSB of the subkey $KI_{4,1,1}$. Let us analyze the function $FI_{4,1}$.

By the structure of the differential, the inputs of $FI_{4,1}$ are of the form

$$\begin{aligned}&\left(X_a^{LL} \oplus KO_{4,1}, X_b^{LL} \oplus KO_{4,1}, X_c^{LL} \oplus KO_{4,1}, X_d^{LL} \oplus KO_{4,1}\right)\\ &\quad = (t, t \oplus 0010_x, t \oplus 0010_x, t),\end{aligned}$$

for some 16-bit value t. After the application of the first S9 of $FI_{4,1}$ the values remain in the form $(t', t' \oplus 0010_x, t' \oplus 0010_x, t')$, for some 16-bit value t', since all four inputs to the S-box S9 are equal. After the XOR and the swap, the values are of the form $(t'', t'' \oplus 2010_x, t'' \oplus 2010_x, t'')$. Hence, the inputs to the first $S7$ are of the form (x, y, y, x), and the outputs of that $S7$ after the XOR with the truncated 9 bits, are of the form (u, v, v, u) (for some 7-bit values v, u). We claim that $u \neq v$ and $u \oplus v \neq 40_x$. Indeed, if we had $u \oplus v = 40_x$, then the 7-bit outputs of the S-box $S7$ had to be of the form $(u', u' \oplus 40_x \oplus 10_x, u' \oplus 40_x \oplus 10_x, u')$. However, the differential $(10_x \to 50_x)$ is impossible for $S7$, and thus this event cannot occur. Similarly, $u = v$ cannot occur since the differential $(10_x \to 10_x)$ is impossible for $S7$.

We now claim that the four 7-bit intermediate values after the XOR with the subkey $KI_{4,1,1}$ are different. Indeed, these values are of the form $(u \oplus k, v \oplus k \oplus 40_x, v \oplus k, u \oplus k \oplus 40_x)$, and these are all different since $u \neq v$ and $u \neq v \oplus 40_x$.

Finally, we consider the S-box $S7$ in the fourth round of $FI_{4,1}$. Its four inputs are all different, and can be divided into two pairs $(u \oplus k, v \oplus k \oplus 40_x)$ and $(v \oplus k, u \oplus k \oplus 40_x)$ with the same difference. Since $S7$ is an *almost perfect nonlinear permutation*,[4] this implies that the two corresponding pairs of outputs have distinct differences, and thus, the XOR of the four output values is necessarily non-zero. Since the XOR of the output values in the right half is zero, we have

$$I_a^1 \oplus I_b^1 \oplus I_c^1 \oplus I_d^1 \neq 0,$$

and hence,

$$X_a^{3L} \oplus X_b^{3L} \oplus X_c^{3L} \oplus X_d^{3L} \neq 0.$$

Therefore, the XOR of the four outputs of $FO4$ is non-zero with probability 1, and since FL is linear and invertible, this implies that the XOR of the four outputs of FL is non-zero with probability 1. This proves that

$$\mathbf{Pr}\big[(X_c \oplus X_d = \beta) \mid (X_a \oplus X_b = \beta) \wedge (Y_a \oplus Y_c = \gamma) \wedge (Y_b \oplus Y_d = \gamma)\big] = 0,$$

and thus the distinguisher fails in this variant of KASUMI, as asserted.

For the sake of completeness, we implemented this variant of KASUMI, and verified experimentally that the number of right quartets with the desired sandwich property was always zero.[5]

[4] An almost perfect nonlinear permutation, introduced in [22], is a permutation $f : GF(2^n) \to GF(2^n)$ such that for any $a \neq 0$, the function $g(x, a) = f(x) \oplus f(x \oplus a)$ assumes exactly 2^{n-1} different values. In an almost perfect nonlinear permutation, for any two pairs of distinct input values with the same difference, the corresponding output pairs cannot have the same difference. The S-boxes $S7$ and $S9$ used in KASUMI were designed as almost perfect nonlinear permutations, in order to obtain maximal security with respect to differential and linear cryptanalysis (see [23]).

[5] We used 100 keys, and for each of them we generated 2^{24} quartets. We first verified using the official key schedule that many right quartets are encountered, and then we modified the key schedule in the way described above. None of the experiments yielded a right quartet.

4.3.2. *A Slight Change in the Differential*

In this example we do not alter the original key schedule of KASUMI, but rather slightly change one of the differentials. Since the considerations we use are similar to the previous example, we present them briefly.

The differential for E_0 remains

$$\alpha = (0_x, 0010\ 0000_x) \to (0010\ 0000_x, 0_x) = \beta,$$

with key difference $\Delta K_{ab} = (0, 0, 8000_x, 0, 0, 0, 0, 0)$. The differential we use for E_1 is slightly changed to

$$\gamma = (0_x, 0100\ 0000_x) \to (0100\ 0000_x, 0_x) = \delta,$$

with key difference $\Delta K_{ac} = (0, 0, 0, 0, 0, 0, 0008_x, 0)$. It is easy to see that the probabilities of the differentials in E_0 and E_1 remain 2^{-2}, like for the original differentials. Also, Eqs. (26), (27), and (28) hold as in the previous example, and hence, in order to show that the probability of the distinguisher is zero, it is sufficient to show that

$$I_a^1 \oplus I_b^1 \oplus I_c^1 \oplus I_d^1 \neq 0. \tag{29}$$

Consider the function $FI_{4,1}$. By the structure of the differentials, its inputs are of the form

$$(t, t \oplus 0010_x, t \oplus 0100_x, t \oplus 0110_x).$$

(Note that unlike the previous example, the four inputs are distinct.) It follows that the inputs to the S-box $S9$ in the first round of $FI_{4,1}$ are of the form (x, x, y, y) (where $x \oplus y = 2_x$) and the inputs to the S-box $S7$ in the second round of $FI_{4,1}$ are of the form (z, w, z, w) (where $z \oplus w = 10_x$). Hence, the corresponding outputs are of the forms (x', x', y', y') and (z', w', z', w'), respectively. Since both these quadruples are *balanced* (i.e., sum up to zero), and there is no key difference in $FI_{4,1}$ (again, unlike the previous example), this implies that in both halves of the intermediate value after the key addition, the quadruples are balanced. Therefore, due to the 4-round Feistel structure, if we show that the outputs of the S-box $S9$ in the third round of $FI_{4,1}$ are *unbalanced*, this will imply that the right half of the output of $FI_{4,1}$ is unbalanced, thus proving that inequality (29) holds.

Consider the four inputs to the S-box $S9$ in the third round of $FI_{4,1}$. By the Feistel structure, they are of the form $(x', x', y', y') \oplus (z, w, z, w) \oplus (KI_{4,1,2}, KI_{4,1,2}, KI_{4,1,2}, KI_{4,1,2})$, and hence, they are balanced. Furthermore, they are distinct, since $z \oplus w = 10_x$, while $x' \oplus y' \neq 10_x$ (since the differential $00000001_2 \to 00001000_2$ is impossible for the S-box $S9$). Since $S9$ is an *almost perfect nonlinear permutation*, this implies that the four outputs are necessarily unbalanced, concluding the proof.

We note that a similar argument holds for almost all choices of modified differentials for E_0 and E_1 in which for one of the differentials the non-zero difference enters the S-box $S9$, and for the other one the non-zero difference enters the S-box $S7$, and shows that the distinguisher must fail. The only two exceptions are:

$$\alpha = (0_x, 0001\ 0000_x), \qquad \gamma = (0_x, 0400\ 0000_x),$$

and:

$$\alpha = (0_x, 0040\ 0000_x), \qquad \gamma = (0_x, 0080\ 0000_x),$$

with appropriately chosen key differences. For these exceptions, the probability r of transition through the middle layer M is close to 2^{-32} (which is the expected probability for a "random" single Feistel round with 64-bit block). For a detailed and experimental analysis of these examples, we refer the reader to [19].

5. Related-Key Attacks on the Full KASUMI

In this section we use the 7-round distinguisher presented in Sect. 4 to devise related-key attacks on the full 8-round KASUMI. Our first attack is a related-key sandwich attack, which requires 2^{26} adaptively chosen plaintexts and ciphertexts encrypted under one of four related keys, and has a time complexity of 2^{32} encryptions. This attack was fully verified experimentally, as described in Sect. 5.1.1. Our second attack is a related-key rectangle-like sandwich attack, which requires 2^{41} chosen plaintexts encrypted under one of four related keys, and has time complexity of 2^{41} encryptions. Although its complexity is higher than that of the first attack, it has the advantage of performing in the more conservative chosen plaintext model (rather than the adaptively chosen plaintext/ciphertext model of the first attack).

5.1. *Related-Key Sandwich Attack on the Full KASUMI*

Our attack on the full KASUMI applies the distinguisher presented in Sect. 4 to rounds 1–7 (see Fig. 8), and retrieves subkey material in round 8. Let $\Delta K_{ab} = (0, 0, 8000_x, 0, 0, 0, 0, 0)$ and $\Delta K_{ac} = (0, 0, 0, 0, 0, 0, 8000_x, 0)$, and let K_a, $K_b = K_a \oplus \Delta K_{ab}$, $K_c = K_a \oplus \Delta K_{ac}$, and $K_d = K_c \oplus \Delta K_{ab}$ be the unknown related keys we wish to retrieve.

The attack algorithm is as follows:

1. *Data Collection Phase:*
   (a) Choose a structure of 2^{24} ciphertexts of the form[6] $C_a = (X_a, A)$, where A is a fixed 32-bit value and X_a assumes 2^{24} arbitrary different 32-bit values. Ask for the decryption of all the ciphertexts under the key K_a and denote the plaintext corresponding to C_a by P_a. For each P_a, ask for the encryption of $P_b = P_a \oplus (0_x, 0010\ 0000_x)$ under the key K_b and denote the resulting ciphertext by C_b. Store the pairs (C_a, C_b) in a hash table indexed by the 32-bit value C_b^R (i.e., the right half of C_b).
   (b) Choose a structure of 2^{24} ciphertexts of the form $C_c = (Y_c, A \oplus 0010\ 0000_x)$, where A is the same constant as before, and Y_c assumes 2^{24} arbitrary different values. Ask for the decryption of the ciphertexts under the key K_c and denote the plaintext corresponding to C_c by P_c. For each P_c, ask for the encryption of $P_d = P_c \oplus (0_x, 0010\ 0000_x)$ under the key K_d and denote the resulting ciphertext by C_d. Then, access the hash table in the entry corresponding to the value $C_d^R \oplus (0_x, 0010\ 0000_x)$, and for each pair (C_a, C_b) found in this entry, apply Step 2 on the quartet (C_a, C_b, C_c, C_d).

[6] We alert the reader that KASUMI employs a swap operation after the last round.

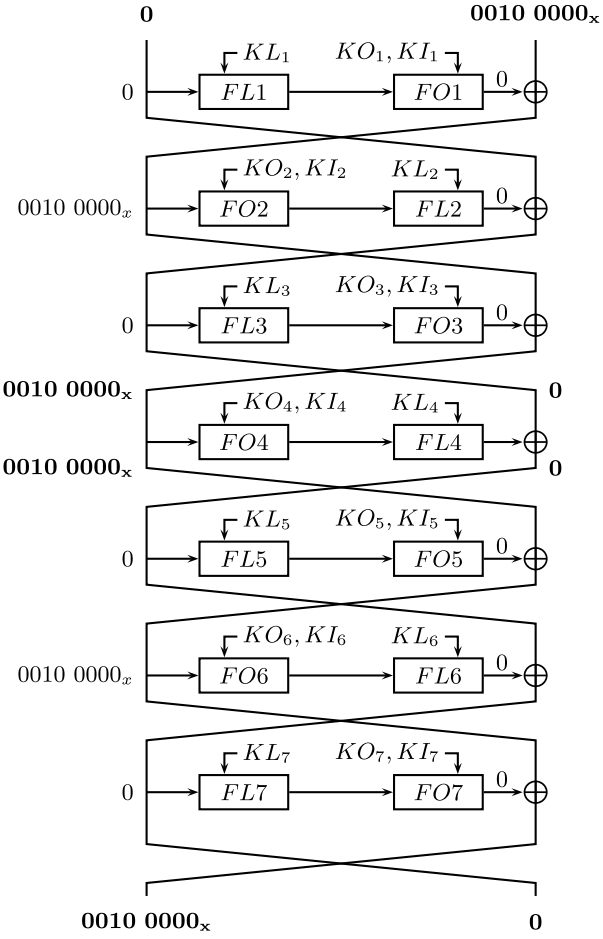

Fig. 8. The 7-round related-key sandwich distinguisher of KASUMI.

In the first step described above, the $(2^{24})^2 = 2^{48}$ possible quartets are filtered according to a condition on the 32 difference bits which are known (due to the output difference δ of the distinguisher), which leaves about 2^{16} quartets with the required differences.

In Step 2 we can identify the right quartets instantly using an extremely lucky property of the KASUMI structure. We note that a pair (C_a, C_c) can be a right quartet only if

$$C_a^L \oplus FL8\big(FO8\big(C_a^R\big)\big) = C_c^L \oplus FL8\big(FO8\big(C_c^R\big)\big), \tag{30}$$

since by the Feistel structure, this is the only case in which the difference after round 7 is the output difference of the sandwich distinguisher (i.e., $\delta = (0010\,0000_x, 0_x)$). However, the values C_a^R and C_c^R are fixed for all the considered ciphertexts, and hence

Table 3. Possible values of $KL_{8,2}$ and $KL_{8,1}$.

OR—$KL_{8,2}$					AND—$KL_{8,1}$				
(X'_{ac}, Y'_{ac})	(X'_{bd}, Y'_{bd})				(X'_{ac}, Y'_{ac})	(X'_{bd}, Y'_{bd})			
	(0, 0)	(0, 1)	(1, 0)	(1, 1)		(0, 0)	(0, 1)	(1, 0)	(1, 1)
(0, 0)	{0, 1}	–	1	0	(0, 0)	{0, 1}	–	0	1
(0, 1)	–	–	–	–	(0, 1)	–	–	–	–
(1, 0)	1	–	1	–	(1, 0)	0	–	0	–
(1, 1)	0	–	–	0	(1, 1)	1	–	–	1

The two bits of the differences are denoted by (input difference, output difference): (X'_1, Y'_1) for one pair and (X'_2, Y'_2) for the other pair.

Eq. (30) yields

$$C_a^L \oplus C_c^L = FL8\big(FO8(A)\big) \oplus FL8\big(FO8\big(A \oplus (0_x, 0010\ 0000_x)\big)\big) = \text{const.} \tag{31}$$

Thus, the value $C_a^L \oplus C_c^L$ is equal for all the right quartets. This allows us to perform the following simple filtering:

2. *Identifying the Right Quartets:* Insert the approximately 2^{16} remaining quartets (C_a, C_b, C_c, C_d) into a hash table indexed by the 32-bit value $C_a^L \oplus C_c^L$, and apply Step 3 only to bins which contain at least three quartets.

Since the probability of a 3-collision in a list of 2^{16} random 32-bit values is $\binom{2^{16}}{3} \cdot 2^{-64} < 2^{-18}$, with very high probability only the right quartets remain after this filtering. The expected number of such quartets is $2^{16} \cdot 2^{-14} = 4$.

In the following step, we treat all the remaining quartets as right quartets. Under this assumption, we know not only the actual inputs to the F-function of round 8, but also the differences between its outputs.

3. *Analyzing Right Quartets:*
   (a) For each remaining quartet (C_a, C_b, C_c, C_d), guess the 32-bit value of $KO_{8,1}$ and $KI_{8,1}$. For the two pairs (C_a, C_c) and (C_b, C_d) use the value of the guessed key to compute the input and output differences of the OR operation in $FL8$ of both pairs.[7] For each bit of this 16-bit OR operation, the possible values of the corresponding bit of $KL_{8,2}$ (given the input and output difference of OR in that bit) are given in Table 3. On average, $(8/16)^{16} = 2^{-16}$ values of $KL_{8,2}$ are suggested by each quartet and guess of $KO_{8,1}$ and $KI_{8,1}$.[8] Since all the right quartets suggest the same key, all the wrong keys are discarded with overwhelming probability, and the adversary obtains the correct value of $(KO_{8,1}, KI_{8,1}, KL_{8,2})$.
   (b) Guess the 32-bit value of $KO_{8,3}$ and $KI_{8,3}$, and use this information to compute the input and output differences of the AND operation in both pairs of

[7] In our case, the guess of $KO_{8,1}$ and $KI_{8,1}$ is sufficient for computing the difference in the left half of the output of $FO8$, since the right half of the input difference to $FO8$ is zero. By the structure of FL, this difference and the output difference of $FL8$ yield the input and output differences of the OR operation.

[8] The simple proof of this claim is given in [7, Sect. 4.3].

Table 4. The number of identified right quartets in 1,000 tests.

Right Quartets	0/1/2	3	4	5	6	7	8	9	10	11	12
Theory (*Poi*(4))	238	195	195	156	104	60	30	13	5	2	0.6
Experiment	247	197	180	167	112	52	30	7	4	3	1

each quartet. For each bit of the 16-bit AND operation of *FL*8, the possible values of the corresponding bit of $KL_{8,1}$ are given in Table 3. On average, $(8/16)^{16} = 2^{-16}$ values of $KL_{8,1}$ are suggested by each quartet and guess of $KO_{8,3}$, $KI_{8,3}$, and thus the adversary obtains the correct value of $(KO_{8,3}, KI_{8,3}, KL_{8,1})$.

4. *Finding the Right Key:* For each value of the 96 bits of ($KO_{8,1}$, $KI_{8,1}$, $KO_{8,3}$, $KI_{8,3}$, $KL_{8,1}$, $KL_{8,2}$) suggested in Step 3, guess the remaining 32 bits of the key, and perform a trial encryption.

The data complexity of the attack is 2^{25} chosen ciphertexts and 2^{25} adaptively chosen plaintexts encrypted/decrypted under one of four keys. The time complexity is dominated by the trial encryptions performed in step 4 to find the last 32 bits of the key, and thus it is approximately equal to 2^{32} encryptions. The probability of success is approximately 76 % (this is the probability of having at least three right pairs in the data pool).

The memory complexity of the attack is also very moderate. We just need to store 2^{26} plaintext/ciphertext pairs, where each pair takes 16 bytes. Hence, the total amount of memory used in the attack is 2^{30} bytes, i.e., 1 GByte of memory.

5.1.1. *Experimental Verification*

We performed two types of experiments to verify our attack. In the first experiment, we just generated the required data, and located the right quartets (thus verifying the correctness of our randomness assumptions). The second experiment was the application of the full attack (both with and without the final exhaustive search over the remaining 32 key bits). All our experiments were carried out on an Intel Core Duo 2 machine with a T7200 CPU (2 GHz, 4 MB L2 Cache, 2 GBytes RAM, Linux-2.6.27 kernel, with gcc 4.3.2 and standard optimization flags (`-O3`, `-fomit-frame-pointers`, `-funroll-loops`), single core, single thread). We recall the fact that the experiment used the official reference implementation of KASUMI from [25], which is not optimized for performance (and thus for exhaustive search).

The first experiment was conducted 1000 times. In each test, we generated the data and found candidate quartets according to Steps 1 and 2 of the attack algorithm. Once these were found, we partially decrypted the quartets, and checked how many quartets were right ones. Table 4 details the outcome of these experiments, which follows the expected distribution.

The second experiment simulated the full attack. We repeated it 100 times, and counted in each case how many times the final exhaustive search over 2^{32} possible

Table 5. The number of exhaustive searches as a function of the number of right quartets (100 experiments).

Right quartets	3	4	5	6	7	8	9	10	Total
Ex. Searches									
2	–	2	8	6	7	3	–	1	27
4	1	6	7	3	2	–	–	–	19
8	1	9	2	–	–	–	–	–	12
16	2	2	4	–	–	–	–	–	8
32	1	2	–	–	–	–	–	–	3
48	1	–	–	–	–	–	–	–	1
64	4	1	–	–	–	–	–	–	5
96	1	–	–	–	–	–	–	–	1
256	1	–	–	–	–	–	–	–	1
512	1	–	–	–	–	–	–	–	1
Total	13	22	21	9	9	3	0	1	78

keys would have been invoked.[9] In 78 out of these 100 experiments, 3 or more quartets were identified to be right ones (the expected number was 76.1), and then the key was found.

About 50 % of the tests were able to identify the right key by invoking either 2 or 4 exhaustive searches. As the first part of the attack (which identifies candidate quartets) takes about 8 minutes, and each exhaustive search (using the official KASUMI source code) takes about 26 minutes, we could find the full 128-bit key in about 50 % of our tests in less than 112 minutes (using a single core). It is important to note that by increasing the running time, one can increase the success rate of the attack without increasing its data requirements. The full distribution of the experiments is given in Table 5.

5.2. *Related-Key Rectangle-Like Sandwich Attack on the Full KASUMI*

The related-key sandwich distinguisher of 7-round KASUMI presented in Sect. 4 can be transformed in a standard way to a related-key rectangle-like sandwich distinguisher in which the probability of a quartet to be a right quartet is $(1/4)^2 \cdot (1/4)^2 \cdot 2^{-64} \cdot 2^{-6} = 2^{-78}$. It is worth noting that in a chosen plaintext manner, one can ensure that the first round of the differential characteristic for E_0 is followed with probability 1 rather than $1/2$, and hence the overall probability can be increased to 2^{-76}. This distinguisher can be used to mount a related-key rectangle-like sandwich attack on the full KASUMI. The attack is very similar to the attack presented in detail in [7], and hence we omit the full description here, and just mention the changes.

Instead of starting with 2^{51} plaintexts in each structure, the adversary can take 2^{39} plaintexts. These plaintexts contain 2^{78} possible quartets, and after the first filtering step only 2^{14} quartets remain. Then in Step 2(a) of the attack, the adversary gets 2^{30} suggestions for 48 key bits (instead of 2^{54} as in [7]), and thus all the wrong suggestions can be discarded (since the right pairs suggest the same value). As a result, the time

[9] The need to perform the final step several times arises since due to the differential nature of the attack, a few key candidates cannot be distinguished from the right key until the exhaustive search step. As can be seen in Table 5, the number of these keys decreases when more right quartets are analyzed.

complexity of the following steps becomes negligible, and the overall time complexity is dominated by the time required to encrypt the 2^{41} chosen plaintexts. This is worse than the 2^{32} time complexity of our sandwich attack but is still practical, and applies in the more realistic chosen plaintext attack model.

As in the ordinary sandwich attack, the memory used during the rectangle-like sandwich attack is dominated by the storage of the plaintexts and the ciphertexts. By first encrypting the data under keys K_a and K_b, storing it in a sorted table, and then encrypting the data under K_c and K_d in a pair-by-pair manner, we have to store only 2^{40} plaintext/ciphertext pairs. Hence, the total memory complexity of the attack is about 16 TBytes (2^{44} bytes). Fortunately, this memory is accessed sequentially and can be relatively slow, so only a few hard disks are needed to store this data.

6. Summary

In this paper we developed a new sandwich attack on iterated block ciphers, and used it to reduce the time complexity of the best known attack on the full KASUMI from an impractical 2^{76} to the very practical 2^{32}. However, the new attack uses both related keys and chosen messages, and thus it is not clear how to apply it in practice to break the specific way in which KASUMI is used in GSM and UMTS (3G) telephony. Our main point was to show that contrary to the assurances of its designers, the transition from MISTY to KASUMI led to a much weaker cryptosystem, which should be avoided in any application in which related-key attacks can be mounted.

6.1. *Future Work*

A drawback in the generic sandwich technique presented in this paper is the lack of rigorous analysis. While in the specific case of KASUMI, we performed a rigorous analysis of the transition probability at the middle slice M and further validated our results with experimental verifications, we were not able to provide such a rigorous analysis for the general case. In particular, we cannot give necessary and sufficient conditions on the cipher structure that ensure that the sandwich attack is applicable, neither we can give explicit conditions under which the independence assumptions the technique relies on are satisfied.

As for conditions that allow mounting a sandwich attack, formulating the exact conditions seems impossible, since such conditions should depend heavily on the exact structure of the cipher. What seems possible is to find other generic structures in which the sandwich attack is applicable (such as the Feistel construction presented in Sect. 2).

As for the independence assumptions, the same problem exists even in the much simpler case of differential cryptanalysis, where one cannot verify whether the independence assumption (known as the hypothesis of stochastic equivalence) holds without a large computational effort. In the case of (related-key) boomerang attacks, where the assumptions are close to the assumptions behind the sandwich attack, such conditions were analyzed in [19,21], and the conclusion was that the assumptions must be checked in each particular case separately. It is likely that the same holds also for the sandwich attack, but any further results regarding the correctness of the independence assumptions in various cases will be interesting.

The last direction for further research refers to the specific case of KASUMI. While the related-key sandwich attack we presented in the paper was accompanied with a full experimental verification, we did not verify experimentally the rectangle-like sandwich attack. An experimental verification of this attack will be interesting, as it will be the first full implementation of a rectangle attack (on any block cipher).

We conclude this paper with a formal statement of the directions for further research raised above.

Problem 1. Find other generic structures in which the sandwich attack is applicable.

Problem 2. Find necessary and sufficient conditions under which the independence assumptions used in the sandwich attack are satisfied.

Problem 3. Verify the validity of the rectangle-like sandwich attack on full KASUMI presented in Sect. 5.2.

Acknowledgements

We would like to thank Steve Babbage and Kaisa Nyberg for their help concerning the status of KASUMI in the GSM/UMTS arena. We would also like to thank Daniel Loebenberger, as well as the referees of both the CRYPTO 2010 version of this paper and of this full version, for their useful comments.

References

[1] A5/1 Security Project, Creating A5/1 rainbow tables (2009). Available online at http://reflextor.com/trac/a51

[2] E. Barkan, E. Biham, Conditional estimators: an effective attack on A5/1, in *Proceedings of Selected Areas in Cryptology 2005*. Lecture Notes in Computer Science, vol. 3897 (Springer, Berlin, 2006), pp. 1–19

[3] E. Barkan, E. Biham, N. Keller, Instant ciphertext-only cryptanalysis of GSM encrypted communication, in *Advances in Cryptology, Proceedings of CRYPTO 2003*. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 600–616

[4] E. Biham, New types of cryptanalytic attacks using related keys. *J. Cryptol.* **7**(4), 229–246 (1994)

[5] E. Biham, O. Dunkelman, N. Keller, New results on boomerang and rectangle attacks, in *Proceedings of Fast Software Encryption 2002*. Lecture Notes in Computer Science, vol. 2365 (Springer, Berlin, 2002), pp. 1–16

[6] E. Biham, O. Dunkelman, N. Keller, Related-key boomerang and rectangle attacks, in *Advances in Cryptology, Proceedings of EUROCRYPT 2005*. Lecture Notes in Computer Science, vol. 3494 (Springer, Berlin, 2005), pp. 507–525

[7] E. Biham, O. Dunkelman, N. Keller, A related-key rectangle attack on the full KASUMI, in *Advances in Cryptology, Proceedings of ASIACRYPT 2005*. Lecture Notes in Computer Science, vol. 3788 (Springer, Berlin, 2005), pp. 443–461

[8] A. Biryukov, C. De Cannière, G. Dellkrantz, Cryptanalysis of SAFER++, in *Advances in Cryptology, Proceedings of CRYPTO 2003*. Lecture Notes in Computer Science, vol. 2729 (Springer, Berlin, 2003), pp. 195–211

[9] A. Biryukov, D. Khovratovich, Related-key cryptanalysis of the full AES-192 and AES-256, in *Advances in Cryptology, Proceedings of ASIACRYPT 2009*. Lecture Notes in Computer Science, vol. 5912 (Springer, Berlin, 2009), pp. 1–18

[10] A. Biryukov, A. Shamir, D. Wagner, Real time cryptanalysis of A5/1 on a PC, in *Proceedings of Fast Software Encryption 2000*. Lecture Notes in Computer Science, vol. 1978 (Springer, Berlin, 2001), pp. 1–18
[11] M. Blunden, A. Escott, Related key attacks on reduced round KASUMI, in *Proceedings of Fast Software Encryption 2001*. Lecture Notes in Computer Science, vol. 2355 (Springer, Berlin, 2002), pp. 277–285
[12] M. Briceno, I. Goldverg, D. Wagner, A pedagogical implementation of the GSM A5/1 and A5/2 "voice privacy" encryption algorithms (1999). Available online at http://cryptome.org/gsm-a512.htm
[13] P. Ekdahl, T. Johansson, Another attack on A5/1. *IEEE Trans. Inf. Theory* **49**(1), 284–289 (2003)
[14] S. Hong, J. Kim, G. Kim, S. Lee, B. Preneel, Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192, in *Proceedings of Fast Software Encryption 1999*. Lecture Notes in Computer Science, vol. 3557 (Springer, Berlin, 2005), pp. 368–383
[15] K. Jia, J. Chen, M. Wang, X. Wang, Practical attack on the full MMB block cipher, in *Proceedings of Selected Areas in Cryptology 2011*. Lecture Notes in Computer Science, vol. 7118 (Springer, Berlin, 2011), pp. 185–199
[16] K. Jia, H. Yu, X. Wang, A meet-in-the-middle attack on the full KASUMI. IACR ePrint report 2011/466
[17] J. Kelsey, B. Schneier, D. Wagner, Key schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES, in *Advances in Cryptology, Proceedings of CRYPTO 1996*. Lecture Notes in Computer Science, vol. 1109 (Springer, Berlin, 1996), pp. 237–251
[18] J. Kim, G. Kim, S. Hong, D. Hong, The related-key rectangle attack—application to SHACAL-1, in *Proceedings of Australasian Conference on Information Security and Privacy 2004*. Lecture Notes in Computer Science, vol. 3108 (Springer, Berlin, 2004), pp. 123–136
[19] J. Kim, S. Hong, B. Preneel, E. Biham, O. Dunkelman, N. Keller, Related-key boomerang and rectangle attacks: theory and experimental analysis. *IEEE Trans. Inf. Theory* **58**(7), 4948–4966 (2012)
[20] M. Matsui, Block encryption algorithm MISTY, in *Proceedings of Fast Software Encryption 1997*. Lecture Notes in Computer Science, vol. 1267 (Springer, Berlin, 1997), pp. 64–74
[21] S. Murphy, The return of the cryptographic boomerang. *IEEE Trans. Inf. Theory* **57**(4), 2517–2521 (2011)
[22] K. Nyberg, Perfect nonlinear S-boxes, in *Advances in Cryptology, Proceedings of EUROCRYPT 1991*. Lecture Notes in Computer Science, vol. 547 (Springer, Berlin, 1991), pp. 378–386
[23] K. Nyberg, L.R. Knudsen, Provable security against differential cryptanalysis, in *Advances in Cryptology, Proceedings of CRYPTO 1992*. Lecture Notes in Computer Science, vol. 740 (Springer, Berlin, 1993), pp. 566–578
[24] 3rd Generation Partnership Project, Technical specification group services and system aspects, 3G security. *Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: KASUMI Specification*, V3.1.1 (2001)
[25] 3rd Generation Partnership Project, Technical specification group services and system aspects, 3G security. *Specification of the A5/3 Encryption Algorithms for GSM and ECSD, and the GEA3 Encryption Algorithm for GPRS; Document 4: Design and Evaluation Report*, V6.1.0 (2002)
[26] D. Wagner, The boomerang attack, in *Proceedings of Fast Software Encryption 1999*. Lecture Notes in Computer Science, vol. 1636 (Springer, Berlin, 1999), pp. 156–170