Advancing the Meet-in-the-Filter Technique: Applications to CHAM and KATAN*

Alex Biryukov [1], Je Sen Teh [1,2], and Aleksei Udovenko [1]

[1] University of Luxembourg {name.surname}@uni.lu
[2] University Sains Malaysia jesen_teh@usm.my

Abstract. Recently, Biryukov et al. presented a new technique for key recovery in differential cryptanalysis, called *meet-in-the-filter* (MiF). In this work, we develop theoretical and practical aspects of the technique, which helps understanding and simplifies application. In particular, we show bounds on MiF complexity and conditions when the MiF-enhanced attack may reach them. We present a method based on trail counting which allows to estimate filtering strength of involved rounds and perform consequent complexity analysis with pen and paper, compared to the computer-aided approach of the original work. Furthermore, we show how MiF can be combined with plaintext structures for linear key schedules, allowing to increase the number of attacked rounds or to reduce the data complexity.
We illustrate our methods on block cipher families CHAM and KATAN and show best-to-date single-key differential attacks for these ciphers.

Keywords: Symmetric-key · Differential cryptanalysis · ARX · NLFSR · CHAM · KATAN

1 Introduction

In over a decade or so, there has been a shift towards portable computing devices such as smart devices and wearable systems. Although these devices have greatly eased the lives of many, they also came with new security challenges, one being the design of cryptographic solutions that are not only efficient and secure, but also have minimal computational requirements (e.g. low memory and energy). Finding a balance between these requirements has been the focus of researchers in the field of *lightweight cryptography*. Many lightweight primitives have been proposed over the years to address this need, including symmetric-key block ciphers such as CHAM [15, 17] and KATAN [5]. Differential cryptanalysis is one of the main techniques for assessing the security of various cryptographic primitives. Resistance to differential attacks has become one of the basic requirements of a modern block cipher. Many variants of the attack have since

* The work was supported by the Luxembourg National Research Fund's (FNR) and the German Research Foundation's (DFG) joint project APLICA (C19/IS/13641232).

been introduced. Recently, a new differential cryptanalysis tool called *meet-in-the-filter* (MiF) was proposed by Biryukov et al. [4]. Using MiF, an attacker can append a large number of rounds for key recovery. When applied to SPECK [2], some of the best differential attacks were reported. However, its application to other block ciphers has yet to be investigated. In this paper, we apply MiF to attack CHAM-64 [15,17] and the KATAN family of block ciphers [5]. The choices of both CHAM and KATAN were motivated by their slow diffusion (slower than SPECK) which allows to append a large number of rounds for key recovery. We report the best differential attacks on both ciphers in the single-key setting using MiF as summarized in Table 1. For a comparison, prior differential attacks on

Table 1: Key Recovery Attacks on CHAM-64 and the KATAN family.

Cipher	Rounds	Time	Data	Memory	Ref.
CHAM-64-128	52	2^{114}	2^{61}	2^{54}	Section 4.2
KATAN-32-80	124	2^{76}	2^{31}	2^{38}	Section 5.3
KATAN-48-80	130	2^{73}	2^{45}	2^{51}	Section 5.4
KATAN-64-80	110	2^{73}	2^{57}	2^{65}	Section 5.5

KATAN are provided in Appendix A.

In addition to new cryptanalysis results for CHAM and KATAN, this paper revisits the practical and theoretical aspects of MiF. We describe theoretical bounds on its complexity and show how simple time complexity estimates can be obtained for MiF-enhanced differential attacks. We also introduce an approach for estimating trail weight distributions which is useful to determine the filtering strength of rounds involved in key recovery, which allows to perform complexity analysis with pen and paper. In addition, we show that MiF can be combined with plaintext structures to increase the number of attacked rounds for ciphers with linear key schedules (e.g. CHAM).

The outline of the paper is as follows. Section 2 provides a brief introduction to the MiF tool. Section 3 present new techniques and theories related to MiF that are used in our attacks. Finally, detailed descriptions of our attacks on CHAM-64 and the entire KATAN family are reported in Section 4 and 5.

2 The Meet-in-the-Filter Technique

A typical differential attack relies on a differential distinguisher over r rounds of a cipher with probability p to which k key-recovery rounds are appended. After guessing the last k round keys, an attacker performs partial decryptions to obtain the output difference after r rounds. If this difference matches the output difference of the r-round distinguisher, then the round key guesses are likely to be correct. Generally, an attacker would try to maximize r to obtain the best attack possible.

MiF [4] is a differential cryptanalysis tool that allows to also maximize k. If a cipher has a relatively slow diffusion an attacker can potentially add a large number of k rounds. MiF produces a set of full k-round trails that are used for key recovery by first splitting k into two parts, $k = s + t$, then processing each part separately to find a meeting point (matching difference in the middle). An illustration of MiF is provided in Figure 1.

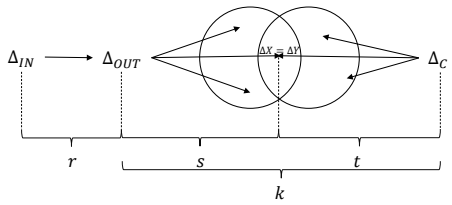

Fig. 1: MiF with an r-round differential and $k = s + t$ key recovery rounds.

Offline, a Matsui-like search is performed in the forward direction over s rounds starting from the output difference of the r-round differential, Δ_{OUT} to an intermediate difference ΔX. The set of these s-round trails (the *MiF cluster*) will be stored. In the online phase, plaintext pairs with difference Δ_{IN} are encrypted over $r + k$ rounds to obtain a ciphertext difference, ΔC. For each ciphertext pair (C_1, C_2), we then perform a reverse search on t rounds in search of a match in the cluster. The set of trails over t rounds is called the *MiF filter*.

Ciphertext pairs that result in a match are candidate *right pairs*, i.e., pairs whose plaintexts have followed the initial r-round differential ($\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}$), and come with a set of suggested k-round trails. The latter will be used in the key-recovery phase which involves guessing round keys and checking if the partial decryption of ciphertext pairs conforms to the suggested trails. Key recovery relies on the following formulation of the Markov assumption:

Assumption 1. *For a differential trail $\Delta P \rightarrow \Delta C$ with a weight w (and possibly truncated intermediate differences), the average* fraction *of pairs of ciphertexts (C_1, C_2) and subkeys for which the partial decryption of (C_1, C_2) follows the trail is equal to 2^{-w}.*

In the end, the attacker is left with a set of candidate keys that are tested with trial decryptions. Optimizing the key recovery phase requires deeper analysis and may vary depending on the target cipher.

3 New Theoretical Analysis of Meet-in-the-Filter Technique and Extensions

3.1 Performance Gain of MiF Key Recovery over Exhaustive Search

We first analyze the theoretical power and limits of MiF with respect to the weights of involved trails or differentials. Indeed, the MiF attack has several parameters and the trade-off between the time and data complexities is not very clear. For the purpose of this subsection, we will assume that the key recovery procedure is perfect: given a trail or even a *differential* over k rounds, it enumerates (without any extra overhead) all the candidate subkeys that satisfy the trail/differential. This is a strong condition and, although it can often be reached in basic MiF attacks, advanced attacks would require more precise analysis of the intermediate costs. We will focus on the following MiF-like setting.

An attacker uses a differential $\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}$ over r rounds with a weight w and queries an $(r+k)$-round encryption of a plaintext pair (P_1, P_2) with $P_1 \oplus P_2 = \Delta_{\mathrm{IN}}$, obtaining a ciphertext pair (C_1, C_2) with $C_1 \oplus C_2 = \Delta C$. The MiF tool suggests a set of valid trails of the form $\Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C$. The attacker may run the perfect key recovery and obtain, by Assumption 1, a list of $2^{K_s - w}$ subkeys, where K_s is the size of the involved subkeys in bits (typically equal to the size of the master key). These subkeys may then be checked using trial decryptions. Questions then arise: What are the chances to hit the correct master key? Should the attacker attempt the key recovery, or, perhaps, it is better to try another pair?

The key insight to answering these questions lies in studying the probability that the right subkey is among the suggested subkeys *posterior to observing the output difference* ΔC. Indeed, if the suggested subkeys are not better than fully randomly guessed subkeys, then the attack is not useful at all. On the other hand, if the suggested subkeys are g times more likely to match the right subkey ($g > 1$), then on average, the attacker would need to test g times fewer subkeys to find the right one, effectively reducing the time complexity (more precisely, of the trial decryption stage) by the factor g. This idea is similar to the classic definition of signal/noise ratio (S/N) by Biham and Shamir [3], and our theory specializes it based on the MiF trail $\Delta_{\mathrm{OUT}} \xrightarrow{r+k} \Delta C$.

In the following, consider two differentials

$$\tau_r = \Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}, \qquad \tau_k = \Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C, \tag{1}$$

with probabilities $\Pr[\tau_r] = p$ and $\Pr[\tau_k] = q$ respectively. Let $\tilde{p}$ be the probability of the full differential $\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C$.

Definition 1. *Define the* gain g *of the pair* (τ_r, τ_k) *as*

$$g = \frac{\Pr[\kappa \textit{ is the right key} \mid \kappa \textit{ satisfies } \Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C]}{2^{-K_s}}, \tag{2}$$

where the probability is over the encryptions of plaintext pairs.

Theorem 1. *The gain g is equal to*

$$g = \frac{\Pr[\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}]}{\Pr[\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C]} = \frac{p}{\tilde{p}}. \tag{3}$$

Proof. For κ to be the right subkey, the encryption must have followed the path $\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C$ (given $\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C$) and κ is one of the subkeys satisfying the transition $\Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C$ (which has expected size $2^{K_s} q$, where q is the transition probability over k rounds). Therefore,

$$g = \frac{\frac{pq}{\tilde{p}} \cdot \frac{1}{2^{K_s} q}}{2^{-K_s}} = \frac{p}{\tilde{p}}. \qquad \square$$

Perhaps counter-intuitively, the proposition shows that the gain does not directly depend on the probability q of trail/differential $\Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C$, except that it is included in the full differential when concatenated with $\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}$. The explanation for that in terms of a differential attack is that *the probability q of the transition τ_k is proportional to the number $2^{K_s} q$ of surviving keys and inversely proportional to the required number $1/(pq)$ of encryptions*: A lower probability transition yields fewer keys to check but requires more encryptions to actually hit it, while a higher probability transition yields more key candidates but happens more often.

The trivial bound $\tilde{p} \geq pq$ translates to the following bound on the gain.

Proposition 1. *The gain g is upper-bounded by $1/q$.*

This is also clear from the fact that the set of (on average) $2^{K_s} q$ subkeys is smaller than all 2^{K_s} keys precisely by the factor of $1/q$. However, this bound is tight only in the case $\tilde{p} = pq$, i.e., when all trails in the differential $\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C$ are going through Δ_{OUT} after the first r rounds. In this case, getting *one* encryption pair with such ΔC is sufficient for the attack to succeed with gain $1/q$, since the right key has to be in those $2^{K_s} q$ suggested keys. In typical attacks, however, the gain would be much smaller. It is as close to $1/q$ as big is the fraction of trails $\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C$ going through Δ_{OUT} (in terms of the total probability).

Usually, we expect all output differences to be equally possible, as described in the following assumption.

Assumption 2. *The probability*[3] *of any differential $\Delta_{\mathrm{IN}} \xrightarrow{r+k} \Delta C$ over the full cipher is equal to $2^{-|C|}$ up to a negligible error, where $|C|$ is the ciphertext size.*

Corollary 1. *Under Assumption 2, the gain g is equal to $2^{|C|} p$.*

[3] Here, we consider probability over all intermediate (long) keys. Therefore, the limitation of a fixed-key permutation to have minimum nonzero differential probability $2^{-|C|+1}$ (over plaintexts/ciphertexts) does not affect this assumption.

Interestingly and counter-intuitively, the gain does not depend on the differential in the MiF part (unless Assumption 2 does not hold and there is a full-round (improbable) differential distinguisher). Note however that here we only consider the final number of key candidates for trial decryptions. Other attack complexities which include the data complexity, the MiF complexity and the intermediate key recovery complexity do depend on the MiF part. For attacks where the trial decryption dominates, the corollary provides a simple time complexity estimate of $2^{K_s}/(2^{|C|}p)$. We further investigate Theorem 1 experimentally[4] in Appendix D.

Remark 1. In principle, the filtering in accordance with the trail does not necessarily need to filter *keys* for each given ciphertext pair: filtering by ciphertext values works as well, since it reduces the total number of trail-subkey candidates.

Remark 2. Our definition of *gain* specializes the S/N ratio to concrete output differences ΔC. Indeed, [3] compute $\mathrm{S/N} = \frac{2^{K_s}p}{w}$, where w is the average number of subkey candidates suggested by a pair (including a possible filtration factor). A given ciphertext difference ΔC defines the differential $\Delta_{\mathrm{OUT}} \xrightarrow{k} \Delta C$ with probability q, so that $w = 2^{K_s}q$ and the S/N value is specialized into simply $\frac{p}{q}$. This is correct as long as the probability of the actual encryption following the differential $\Delta_{\mathrm{IN}} \xrightarrow{r} \Delta_{\mathrm{OUT}}$ is equal to p; in particular, this is true if we average the S/N value over all possible ciphertext differences. However, for a given ciphertext difference, the actual probability is equal to $\frac{pq}{\tilde{p}}$ (see above), yielding the posterior factor $\frac{q}{\tilde{p}}$. Note that even Assumption 2 (fixing $\tilde{p}$ to $2^{|C|}$) does not make the gain match the S/N definition, since the MiF trails defining the probability q still vary depending on the observed ciphertext difference (but the gain will average to S/N over all possible differences). We conclude that the gain theory is more fine-grained than the S/N formula of [3], giving more insights into the MiF attack.

Trails or differentials? In order to provide final complexity estimates for the trial decryption step, there are two cases depending on whether a trail is used for an attack or a differential. We will assume that the recovered subkey can be used to obtain all other subkeys using the cipher's decryption procedure. If only a part of the subkey is recovered, the rest can be recovered by exhaustive search.

In a *differential*-based attack, the recovered subkey is simply used to decrypt one r-round (partially decrypted) ciphertext and checked against the known plaintext. We assume the cost of this is equal to r rounds of the primitive, or, $r/(r+k)$ full primitive decryptions.

In a *trail*-based attack, a surviving pair can be decrypted round-by-round and checked for conformance to the trail. The expected number of round decryptions can be computed by using the trail's round weights $(w_1, w_2, \ldots, w_r)$ as

$$c = 1 + 2^{-w_r} \cdot (1 + 2^{-w_{r-1}} \cdot (\ldots)). \tag{4}$$

[4] Codes and other relevant information are available at github.com/cryptolu/MeetInTheFilter_CHAM_KATAN.

Since both values have to be decrypted to test the difference, the cost has to be doubled so that the final cost is equal to $2c/(r+k)$ full primitive decryptions.

Since the final number of candidates is proportional to the probability of the trail/differential, the final complexity can be expressed as

$$\min\left(\frac{2c}{r+k}\cdot\frac{1}{p_{\text{trail}}},\ \frac{r}{r+k}\cdot\frac{1}{p_{\text{diff}}}\right)$$

full primitive decryptions.

We remark that a mix of the two methods is possible by fixing several final rounds of a differential to the same trail. The associated cost c can be computed as in (4), applied to the last $t \le r$ rounds with the additional cost of $(r-t)2^{-w_r-w_{r-1}-\ldots-w_{r-t+1}}$ single-round decryptions (i.e., for a pair surviving all t last rounds of the trail, we need to decrypt only one text up to the plaintext to check). We will use this method in the attack on CHAM (Section 4.2).

3.2 MiF-like Key Recovery Applied to Plaintext Structures

When the attacked cipher has a simple (e.g., linear) key schedule, MiF-like key recovery can be also performed at the first round of the cipher, combined with a standard technique in differential cryptanalysis - *plaintext structures*. The latter allows to construct a compact set of plaintexts containing many pairs satisfying one of the given differences.

The standard approach is to start with a single difference Δ_{IN}, propagate it backward by a few rounds in all possible ways to determine the set of possibly active bits, and construct a structure consisting of plaintexts with active bits taking all possible values and inactive bits set to any constant. After the whole structure is encrypted, the attacker enumerates all pairs in the structure and analyzes conditions on which the pair would reach the difference Δ_{IN} after the initial rounds. Typically, there is a strong filter quickly discarding many such pairs. If the main differential $\Delta_{\text{IN}} \xrightarrow{r} \Delta_{\text{OUT}}$ requires more pairs, the process is repeated for the same structure but using different constants for inactive bits.

We recall that the core idea of MiF is to find a trail connecting the differences, and to use it to recover candidates for the intermediate subkey bits. This methodology can be directly applied to plaintext structures as well. Since the attacker needs to bound the activity pattern after propagating Δ_{IN} backward, we will assume that all t trails over these initial rounds can be explicitly enumerated. Then, during the attack, the attacker would simply enumerate these trails and choose accordingly pairs from the structure instead of enumerating all pairs. Note that a structure with n active bits contains 2^n plaintexts and distinct 2^{n-1} pairs of plaintexts satisfying a chosen difference fitting the pattern. Therefore, $t2^{n-1}$ pairs are enumerated instead of all 2^{2n-1} pairs, per one structure.

Then, for each plaintext-trail pair, the attacker can also apply MiF-like key recovery in addition to the ciphertext-side key recovery. This stage can in fact be precomputed offline. A necessary constraint is that it should be possible to combine the subkey bits recovered from the plaintext and ciphertext sides, which is typically the case for linear key schedules (at a reasonable cost).

3.3 Computing or Estimating the Average Trail Probability

We propose the following very simple but powerful theorem that relates the average probability of a trail to the total number of trails, where we only consider trails starting with a fixed difference.

Theorem 2. *Let Δ be a state difference of a cipher. Let T be the set of all possible l-round trails starting at Δ. Then, the average probability of a trail from T is equal to $1/|T|$.*

Proof. Follows from the fact that all trails starting from a single fixed difference must have probabilities summing to 1. □

For ARX ciphers, counting the number of valid output differences for a single ADD operation can be done efficiently using bit-based dynamic programming.

Lemma 1. *Let α, β be fixed differences for the n-bit inputs of ADD or SUB. Then, the number of differences γ such that $(\alpha, \beta) \to \gamma$ is a valid differential transition through the chosen operation can be computed in time $\mathcal{O}(n)$.*

Proof. The idea is to iterate an index i from the least significant bit to the most significant bit and keep 3 counters for the numbers of differences γ (defined up to the current bit) such that: (a) $\alpha_i = \beta_i = \gamma_i = 0$, (b) $\alpha_i = \beta_i = \gamma_i = 1$, (c) $\neg(\alpha_i = \beta_i = \gamma_i)$. Since the new bit of γ is defined by one of the three cases, and all bits of α and β are given, it is easy to update the counters. In particular, case (c) creates both possibilities for the new difference bit (branching factor 2), while cases (a) and (b) define the new difference bit deterministically. See Algorithm 1 for details. □

Algorithm 1 Counting differential trail extensions through single ADD or SUB

Input: $\alpha, \beta \in \mathbb{F}_2^n$
Output: $|\{\gamma \in \mathbb{F}_2^n \mid (\alpha, \beta) \to \gamma \text{ is valid through ADD/SUB}\}|$

if $\alpha_0 = \beta_0 = 0$ **then**
  $(c_0, c_1, c_{\neq}) \leftarrow (1, 0, 0)$
else
  $(c_0, c_1, c_{\neq}) \leftarrow (0, 0, 1)$
end if
for $i \in \{1, \ldots, n-1\}$ **do**
  if $\alpha_i = \beta_i = 0$ **then**
    $(c_0, c_1, c_{\neq}) \leftarrow (c_0 + c_{\neq}, 0, c_1 + c_{\neq})$
  else if $\alpha_i = \beta_i = 1$ **then**
    $(c_0, c_1, c_{\neq}) \leftarrow (0, c_1 + c_{\neq}, c_0 + c_{\neq})$
  else if $\alpha_i \neq \beta_i$ **then**
    $(c_0, c_1, c_{\neq}) \leftarrow (0, 0, c_0 + c_1 + 2c_{\neq})$
  end if
end for
return $c_0 + c_1 + c_{\neq}$

Theorem 2 and Lemma 1 allow to compute the average trail probability for a given cipher and a chosen difference for a relatively large number of rounds. The idea is to explicitly enumerate all possible trails for l rounds and then use Algorithm 1 on each trail to extend by 1 or more rounds implicitly, depending on the cipher's structure. For example, Speck allows only 1-round extension in both directions, while CHAM allows 3-round extension forwards and 1-round extension backwards. Furthermore, typically, after a few rounds, the round's branching factor (multiplier to the number of trails) converges to the branching factor of a random difference transition which is usually known (for example, it is about $2^{12.1}$ for a 16-bit ADD [4]). The latter estimation method is relevant also for non-ARX primitives, for which Algorithm 1 is non-applicable. Therefore, these techniques allow to count or to estimate the number of trails over any number of cipher's rounds.

3.4 Estimating the Truncated Trail Weight Distribution

For complexity analysis, we need to compute the probability of each round's differential transition, averaged over a given set of trails. For this, we employ the following assumption.

Assumption 3. *Let T_i be the set of all trails over i rounds of a cipher. Then, the probability of differential transitions at round $j \le k$ averaged over the trails from T_k can be estimated as $|T_{j-1}|/|T_j|$, letting $|T_0| = 1$.*

Example 1. In CHAM64, the difference $(\mathtt{2000}, \mathtt{1000}, \mathtt{2810}, \mathtt{0020})$ spans forward $2^{19.55}$ trails over 4 rounds and $2^{29.89}$ trails over 5 rounds. Therefore, we assume that the average probability of a differential transition over the 5th round is equal to $2^{-10.34}$ (see Table 2b).

The intuition for this assumption is based on Theorem 2. Indeed, the average probability of all trails over the first $j-1$ rounds is equal to $1/|T_{j-1}|$, and over the first j rounds it is equal to $1/|T_j|$. Since for a single trail the probabilities over rounds are multiplied, it is natural to use such a product rule for *average* probabilities of trails in order to estimate the single round's average probability. The possibility of approximation error comes from the fact that extending trails over j rounds to k rounds may change the distribution of the j-round prefixes (by changing their multiplicities).

Furthermore, the approximation error is limited by the fact that the average round probabilities computed in such way do multiply to the correct average trail probability $1/|T_k|$. Therefore, the actual average probabilities may only shift filtration power from one round to another, unlikely to significantly affect complexity analysis of our attacks. Our experiments on KATAN-32 in Appendix E show that estimates using Assumption 3 closely approximates the actual average trail weights.

Eichlseder and Kales [8] adopted a similar approach to estimate the differential probability for semi-truncated differential characteristics. Rather than

enumerating all possible trails, they calculate the sum of probabilities of all compatible differential characteristics for a fixed input difference, averaged over all compatible input differences.

4 Cryptanalysis of Round-reduced CHAM

4.1 CHAM Revisited

CHAM is a family of lightweight block ciphers based on the ARX construction [15, 17]. It consists of 3 members, CHAM-64-128 (to which we refer to as CHAM-64 in the rest of the paper), CHAM-128-128 and CHAM-128-256 with 88, 112 and 120 rounds respectively, where CHAM-n-k refers to a variant with an n-bit block and k-bit secret key. Each block is processed as $m = \frac{n}{4}$-bit words using three main operations: bitwise XOR, addition modulo 2^m and bitwise rotation. The two consecutive rounds of CHAM are depicted in Figure 2. CHAM has a linear key schedule that generates $\frac{2k}{m}$ m-bit words defined as

$$RK[i] = K[i] \oplus ROL_1(K[i]) \oplus ROL_8(K[i]), \tag{5}$$

$$RK[i + \frac{k}{m} \oplus 1] = K[i] \oplus ROL_1(K[i]) \oplus ROL_{11}(K[i]). \tag{6}$$

Note that the master key words can be calculated from round subkeys by inverting these linear maps, for example, by precomputed lookup tables.

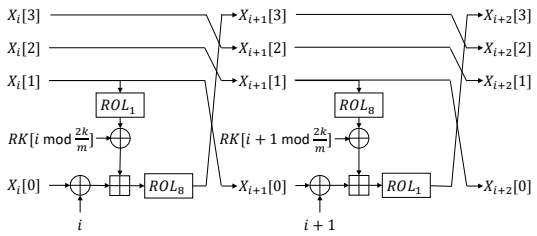

Fig. 2: Two consecutive rounds of CHAM starting from an even i-th round

The following observations on CHAM are used in our attack:

Observation 1. *In any 4 rounds of* CHAM*, the input and the output difference together determine the full 4-round differential trail (efficiently). Consequently, 4 rounds of* CHAM *do not have any trail clustering.*

Observation 2. *The addition in the i-th round of* CHAM *takes as inputs the outputs of additions in the $(i-3)$-rd and $(i-4)$-th rounds. The subtraction (in decryption) in the i-th round of* CHAM *takes as inputs the outputs of the subtractions in the $(i+1)$-st and $(i+4)$-th rounds.*

Prior differential cryptanalysis findings on CHAM only involved identifying the best differential trails. Apart from attacks briefly described by the designers, no other key recovery attacks have been proposed so far.

Appendix B provides some details of the CHAM differentials used in our paper. While experimentally verifying the validity of these differentials, we also found that CHAM's differentials are highly key-dependent. This key-dependency is further investigated in Appendix C.

4.2 Attack on 52-round CHAM-64

We use a differential over 40 rounds of CHAM having probability $p = 2^{-60.05}$ (with the first and last 4 rounds having fixed differences), and an attack split $4+40+4+4$. The main 40-round differential can be extended 4 rounds backward in $2^{35.67}$ possible ways (see Table 2a), with 10 least significant bits of the last word never active in the plaintext difference. This list of differences can be precomputed.

We encrypt $2^{7.05}$ structures of 2^{54} plaintexts ($2^{61.05}$ encryptions). After extracting pairs with one of $2^{35.67}$ possible plaintext differences in each structure, we obtain $2^{61.05+35.67-1} = 2^{95.72}$ plaintext-ciphertext pairs for analysis, each accompanied by a candidate trail in the first $4 + 40$ rounds. Note that we expect to find 1 right pair following the main 40-round differential in this set (by the linearity of expectation), since the sum of probabilities of all trails in the 4 prefix rounds is equal to 1, and we consider $2^{60.05}$ such sets. Each such pair is processed by MiF, getting a list of candidate trails for the last 8 rounds. From Table 2b we can see that each ciphertext difference induces $2^{64.84-64} = 2^{0.84}$ trails[5] on average, resulting in $2^{96.56}$ full 52-round trails (with associated plaintext and ciphertext pairs) for analysis. This can be done by expanding the difference Δ_{OUT} forward by $2^{19.55}$ possible 4-round trails and checking the validity of differential transitions through ADD in the bottom 4 rounds, leading to a time complexity of about $2^{19.55}$ one-round encryptions of CHAM per pair ($2^{116.11}$ total).

Using methodology from Section 3.3, we computed the filtering probabilities for each round of the first and last 8 rounds, see Table 3 (computed from Table 2a and Table 2b). Now, we will guess the subkeys in a carefully chosen order, cross-checking the top and bottom subkeys as soon as possible. The order must satisfy Observation 2, namely, guessing or verifying round r at the top must have rounds $r-3, r-4$ guessed (can skip 2 rounds); guessing or verifying round r at the bottom must have rounds $r+1, r+4$ guessed (cannot skip rounds). The timeline of the procedure is given in Table 4.

The procedure uses 2 main actions: computing a representation of the set of candidates for a subkey of one of the rounds; filtering a set of candidates for some subkey by a round from the other side. Both filters are based on one known incoming value into the addition/subtraction and the known differential transition.

[5] When extending 8 rounds forward, there are $2^{64.84}$ possible trails. Since there are only 2^{64} possible ciphertext differences, each ciphertext difference suggests $2^{0.84}$ trails on average.

Table 2: CHAM-64 trail statistics. Weight is defined as the $-\log_2$ of the probability.

(a) Backward extension from difference $\Delta_{\text{IN}} = (0020, 0010, 1020, 2800)$.

Round	#Trails	Avg. Weight (this round)
-1	$2^{4.17}$	4.17
-2	$2^{11.89}$	7.72
-3	$2^{23.8}$	11.91
-4	$2^{35.67}$	11.87

(b) Forward extension from difference $\Delta_{\text{OUT}} = (2000, 1000, 2810, 0020)$.

Round	#Trails	Avg. Weight (this round)
1	$2^{1.58}$	1.58
2	$2^{8.12}$	6.54
3	$2^{15.46}$	7.34
4	$2^{19.55}$	4.09
5	$2^{29.89}$	10.34
6	$2^{39.95}$	10.06
7	$2^{52.57}$	12.62
8	$2^{64.84}$	12.27

Table 3: Filter strength (top and bottom 8 rounds) in the 52-round attack on CHAM-64. Rounds 1-4 correspond to the 4-round backwards extension at the top; rounds 5-8 correspond to the first 4 rounds of the main differential trail; rounds 45-52 correspond to the 8-round forward extension (MiF) at the bottom.

Master key word	Round	Subkey	Filter	Round	Subkey	Filter	Total
$K[0]$	1	$RK[0]$	$2^{-11.87}$	49	$RK[0]$	$2^{-10.34}$	$2^{-22.21}$
$K[1]$	2	$RK[1]$	$2^{-11.91}$	50	$RK[1]$	$2^{-10.06}$	$2^{-21.97}$
$K[2]$	3	$RK[2]$	$2^{-7.72}$	51	$RK[2]$	$2^{-12.62}$	$2^{-20.34}$
$K[3]$	4	$RK[3]$	$2^{-4.17}$	52	$RK[3]$	$2^{-12.27}$	$2^{-16.44}$
$K[4]$	5	$RK[4]$	$2^{-1.00}$	46	$RK[13]$	$2^{-6.54}$	$2^{-7.54}$
$K[5]$	6	$RK[5]$	$2^{-2.00}$	45	$RK[12]$	$2^{-1.58}$	$2^{-3.58}$
$K[6]$	7	$RK[6]$	$2^{-3.00}$	48	$RK[15]$	$2^{-4.09}$	$2^{-7.09}$
$K[7]$	8	$RK[7]$	$2^{-2.00}$	47	$RK[14]$	$2^{-7.34}$	$2^{-9.34}$
all	1-8	RK[0-7]	$2^{-43.68}$	43-50	RK[0-3,12-15]	$2^{-64.84}$	$2^{-108.52}$

Attack Complexity. The final time complexity (based on Table 4) is dominated by $2 \cdot 2^{117.63}$ one-round key recovery analyses per candidate subkey, and the cost to verify the $2^{116.05}$ final subkey candidates. We assume a 2-round cost for enumerating a subkey candidate. Since the last 4 rounds of the differential are fixed to a trail with round weights $(1, 2, 3, 2)$, we can test a key candidate with

$$1 + 2^{-2}(1 + 2^{-3}(1 + 2^{-2}(1 + 2^{-1})))$$

one-round decryptions on average and 2^{-8} 40-round decryptions, totalling to $1.45 = 2^{0.54}$ one-round decryptions. We obtain the final estimation of

$$2^{116.05+0.54} + 2^{118.63} \times 2 = 2^{119.80}$$

Table 4: Guessing procedure in the 52-round attack on CHAM-64. Time is measured in one-round key recovery analysis cost per key candidate.

Step	Guess subkey	Verify subkey	Filter	Time	Trail-key pairs remaining
0	initial (after MiF)				$2^{96.56}$
1	R52 : $K[3]$		$2^{-12.27}$	$2^{100.29}$	$2^{100.29}$
2	R51 : $K[2]$		$2^{-12.62}$	$2^{103.67}$	$2^{103.67}$
3		R3 : $K[2]$	$2^{-7.72}$	$2^{103.67}$	$2^{95.95}$
4	R2 : $K[1]$		$2^{-11.91}$	$2^{100.04}$	$2^{100.04}$
5		R50 : $K[1]$	$2^{-10.06}$	$2^{100.04}$	$2^{89.98}$
6	R1 : $K[0]$		$2^{-11.87}$	$2^{94.11}$	$2^{94.11}$
7		R49 : $K[0]$	$2^{-10.34}$	$2^{94.11}$	$2^{83.77}$
8		R4 : $K[3]$	$2^{-4.17}$	$2^{83.77}$	$2^{79.60}$
9	R48 : $K[6]$		$2^{-4.09}$	$2^{91.51}$	$2^{91.51}$
10		R7 : $K[6]$	$2^{-3.00}$	$2^{91.51}$	$2^{88.51}$
11	R47 : $K[7]$		$2^{-7.34}$	$2^{97.17}$	$2^{97.17}$
12	R46 : $K[4]$		$2^{-6.54}$	$2^{106.63}$	$2^{106.63}$
13		R5 : $K[4]$	$2^{-1.00}$	$2^{106.63}$	$2^{105.63}$
14		R8 : $K[7]$	$2^{-2.00}$	$2^{105.63}$	$2^{103.63}$
15	R6 : $K[5]$		$2^{-2.00}$	$2^{117.63}$	$2^{117.63}$
16		R45 : $K[5]$	$2^{-1.58}$	$2^{117.63}$	$2^{116.05}$

one-round encryptions, equal to $2^{114.10}$ 52-round CHAM encryptions. The data complexity is $2^{61.05}$ chosen-plaintext encryptions. The 2^{54} 64-bit blocks required to store plaintexts in a structure dominate memory complexity.

5 Cryptanalysis of Round-reduced KATAN

5.1 KATAN Revisited

The KATAN family of block ciphers comprises three variants denoted as KATAN-b, where the block size b is 32, 48 or 64 [5]. KATAN consists of two nonlinear feedback shift registers (NLFSR) that store and update the plaintext and an LFSR to generate round subkeys. All variants of KATAN have an 80-bit key, the same key schedule and 254 rounds. They differ by register lengths, bit positions that enter the feedback functions, and the number of *steps*, which is the number of times the round function is repeated using the same subkey each round. Figure 3 depicts one round of KATAN where L_1 and L_2 are two NLFSRs while k_a and k_b are two round key bits. Given an 80-bit master key K, the subkey of round i is $k_a||k_b = k_{2\cdot i}||k_{2\cdot i+1}$ where

$$k_i = \begin{cases} K_i, & \text{for } i = 0, \ldots, 79, \\ k_{i-80} \oplus k_{i-61} \oplus k_{i-50} \oplus k_{i-13}, & \text{otherwise.} \end{cases}$$

The L_1 and L_2 registers are updated by two nonlinear feedback functions, f_b and f_a respectively, defined as follows:

$$\begin{aligned} f_a(L_1) &= L_1[x_1] \oplus L_1[x_2] \oplus (L_1[x_3] \wedge L_1[x_4]) \oplus (L_1[x_5] \wedge IR) \oplus k_a, \\ f_b(L_2) &= L_2[y_1] \oplus L_2[y_2] \oplus (L_2[y_3] \wedge L_2[y_4]) \oplus (L_2[y_5] \wedge L_2[y_6]) \oplus k_b, \end{aligned} \tag{7}$$

where IR is the irregular update bit depending on the round. The selection of x_i and y_i bits differ for each variant of KATAN. During each *step*, LSBs of L_1 and L_2 are updated by f_b and f_a respectively. For KATAN-48 and -64, each round has two and three steps respectively. Table 5 defines the register lengths and bit positions that enter the feedback functions for all variants.

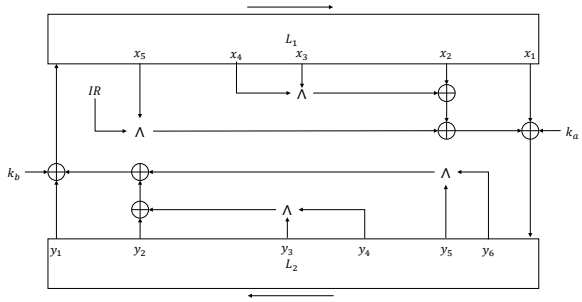

Fig. 3: One round of KATAN

The most successful attacks on all variants of KATAN are multidimensional meet-in-the-middle attacks that span up to 206, 148 and 129 rounds for KATAN-32, 48 and 64 respectively [16]. As the goal of this paper is to use MiF to improve

Table 5: Parameters for the KATAN-b family of block ciphers

b	$\lvert L_1\rvert$	$\lvert L_2\rvert$	x_1	x_2	x_3	x_4	x_5	y_1	y_2	y_3	y_4	y_5	y_6
32	13	19	12	7	8	5	3	18	7	12	10	8	3
48	19	29	18	12	15	7	6	28	19	21	13	15	6
64	25	39	24	15	20	11	9	38	25	33	21	14	9

differential cryptanalysis, we will compare our results against the best single-key and related-key differential attacks as summarized in Table 7 of Appendix A.

In the single-key setting, Albrecht and Leander proposed a 115-round attack with time complexity $T = 2^{78}$ that exploits the full difference distribution of KATAN-32 [1]. Their attack was not computationally feasible for larger variants. A 117-round amplified boomerang attack with $T = 2^{79.3}$ was later introduced by Chen et al. [6]. The best single-key differential attacks on KATAN-48 and 64 were reported by Knellwolf et al. [13] using an approach known as conditional differential cryptanalysis. The latter is applicable to ciphers with NLFSR-based constructions. Using this approach, around 2 bits of subkey information could be recovered after 70 and 68 rounds of KATAN-48 and KATAN-64 respectively.

5.2 Key Recovery Observations

The differential propagation through AND is the basic block for constructing trails over the KATAN structure.

Fact 1. *Let $(\alpha, \beta) \to \gamma$ be a valid differential transition through AND (2-to-1 bit). Then,*

1. *$\alpha = \beta = 0$ implies $\gamma = 0$ (probability 1);*
2. *otherwise, the transition has probability 1/2 (for both cases $\gamma = 0$ and $\gamma = 1$).*

The structure of KATAN's step allows to easily derive the average branching factor of differential trails, i.e., how many 1-step trails does a random difference span on average. This is useful for estimating the MiF complexity. Note that this factor does not depend on the version of KATAN.

Proposition 2. *Let α be a uniformly random state difference in* KATAN*. Then, it is expected to span $217/64 \approx 2^{1.76}$ 1-step trails (in any chosen direction).*

Proof. The L_2 register has 2 ANDs applied to it, but their outputs are XORed together. The transition we consider includes the full step and does not specify the intermediate output difference of the ANDs. The output difference of the two XORed ANDs is either fully determined (when both ANDs are inactive, i.e., have zero difference in all input and output bits), which happens with probability 1/16, or can be equal to 0 or 1, which happens with probability 15/16. For the single AND applied to the L_1 register, the situation is similar but simpler: the transition has a single extension if the AND is inactive (happens in 1/4 of the

cases), or 2 extensions otherwise (in 3/4 of the cases). We obtain the expected number of 1-step extensions is equal to

$$\frac{15}{16}\cdot\frac{3}{4}\cdot 4+\frac{1}{16}\cdot\frac{3}{4}\cdot 2+\frac{15}{16}\cdot\frac{1}{4}\cdot 2+\frac{1}{16}\cdot\frac{1}{4}\cdot 1=\frac{217}{64}. \qquad \square$$

Our experiment on KATAN-32 in Appendix E supports the validity of Proposition 2.

The key recovery behaviour of AND (with respect to a given differential transition) is illustrated in Table 6a and Table 6b. The tables show that an active differential transition through AND is conditioned by an affine function of the involved *values*. More precisely, if one of the input bits is active, then we learn the value of the other (inactive) bit (which has to be the same for both texts since it is inactive). If both input bits are active, then we learn the value of the difference between the input bits (which, again, has to be the same for both texts even though they have to be different).

Table 6: Differential properties of AND.

(a) Output differences $\Delta(xy)$ of transitions through AND.

(x,y)	$(\Delta x,\Delta y)$ $(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
$(0,0)$	0	0	0	1
$(0,1)$	0	0	1	0
$(1,0)$	0	1	0	0
$(1,1)$	0	1	1	1

(b) Conditions and output values induced by differential transitions through AND.

Diff. transition $(\Delta x,\Delta y)\to\Delta(xy)$	Condition	Output xy
$(0,0)\to 0$	-	xy
$(0,1)\to 0$	$x=0$	0
$(0,1)\to 1$	$x=1$	y
$(1,0)\to 0$	$y=0$	0
$(1,0)\to 1$	$y=1$	x
$(1,1)\to 0$	$x\oplus y=1$	0
$(1,1)\to 1$	$x\oplus y=0$	$x=y$

Proposition 3. *In the encryption mode, the key addition k_b (resp. k_b) going into the L_1 (resp. L_2) register does not affect an AND operation during x_4+1 (resp. y_6+1) encryption steps. The concrete numbers of steps are* 6/8/12 *(resp.* 4/7/10*) for 32-/48-/64-bit versions of* KATAN.

Proposition 4. *In the decryption mode, the key addition k_a going into the L_1 register affects an AND operation only after x_1-x_3 decryption steps*[6]*, equal to*

[6] In a decryption step, due to the register shift, the taps $x_2,\ldots,x_5,y_2,\ldots,y_6$ are increased by 1 in order to match the same bits used in the encryption. This explains why Proposition 3 has an extra step compare to Proposition 4.

4/3/4 *for 32-/48-/64-bit versions of* KATAN *respectively. Similarly, the key addition* k_b *going into the* L_2 *register affects an AND operation only after* $y_1 - y_3$ *decryption steps, equal to* 6/7/5 *for 32-/48-/64-bit versions of* KATAN *respectively.*

In our attacks, we guess key bits only when they are input to an AND being currently decrypted (bits x_3 and y_3). Note that the XOR feed takes as input a bit placed after the first AND input bit. In this way, we always guess concrete subkey bits and not linear functions of them.

5.3 Attack on 124-round KATAN-32

Appendix B provides details of the KATAN differentials used in our attacks. We attack rounds 124 rounds of KATAN-32 after skipping the first 29 rounds (i.e., rounds 30-153). This skip is motivated by rounds 34-107 having a local minimum in the fraction of rounds with IR=1, leading to slower diffusion. The rounds are split as follows:

- Rounds 30-33 (4): free rounds at the top (key-independent transitions);
- Rounds 34-107 (74): main differential(s): `40028200` → `21000004/21000006`;
- Rounds 108-149 (42): MiF and key recovery (involves 80 distinct subkey bits);
- Rounds 150-153 (4): free rounds at the bottom (key-independent transitions).

Using the trail(s). The two best 74-round differential trails covering rounds 34-107 both have the same weight 31 and differential probability $2^{-30.23}$. Since the differential effect is weak, we choose to use the best *trail* (rather than differentials), as it allows to verify the subkey candidates round-by-round by checking conformance to the trail (thus avoiding full cipher decryption). This allows to save a few bits in the time complexity since the number of rounds is large.

Free rounds. We choose 2^{30} random inputs pairs at round 34, conforming to the difference `40028200`. Each value is then decrypted by 4 rounds using zero subkey bits. These bits do not affect the differential propagation, so that the difference `40028200` will be satisfied at round 34 during actual encryption with real subkey bits. These 2^{30} pairs are than encrypted by the oracle.

Similarly, the last 4 rounds can be decrypted without involving the key material. Although the rounds actually perform key additions, they are effectively delayed until the first AND operation, which takes 4 rounds for the L1 register and 6 rounds for the L2 register.

MiF Trail Enumeration. We effectively end up with 2^{30} pairs of encryptions covering rounds 34-149, candidates for satisfying one of the two differentials in rounds 34-107.

For each of the two differences Δ_{OUT}, we run the MiF procedure and generate possible trails connecting Δ_{OUT} and the (partially decrypted) ciphertext

difference. For each trail, we run the key recovery procedure (described below). The time complexity of MiF can be estimated as follows. By exhaustive trail enumeration, we observe that the two differences span respectively $2^{30.36}$ and $2^{32.87}$ trails ($2^{33.10}$ total) over 23 rounds (108-130). These trails form the MiF cluster. Then, for each ciphertext difference, we enumerate all possible trails backwards over 19 rounds (131-149), leading to $(2^{1.76})^{19} = 2^{33.44}$ trails on average (by Proposition 2)[7]. Each of the obtained differences is checked against the cluster. This requires $2^{30} \times 2^{33.44} = 2^{63.44}$ lookups (each may return several trails), which is negligible compared to the key recovery complexity (see below).

For the record, we expect to check in total over $2^{30+64.46-32} + 2^{30+66.97-32} = 2^{65.20}$ trails. Here, $2^{64.46}$ and $2^{66.97}$ are the number of 42-round trails spanned by the two chosen differences respectively (computed iteratively using dynamic programming).

Key Recovery. Each generated trail is passed through the basic round-by-round key recovery procedure, using the associated trail as a filter. By the gain analysis (Section 3.1), we expect the final number of key candidates to be around $2^{80-(32-\Pr T)} = 2^{79}$, which provides an estimate for the attack's complexity, given that the available differential filter can be efficiently used. This can be ensured by the simple structure of the cipher. Indeed, the total number of key candidates is only growing with the recursion depth, since the average trail probability per step is $2^{-1.76}$ and 2 subkey bits are involved per step, yielding an expansion factor $2^{0.24}$ per step (after the first few key recovery rounds where subkey material use is sparse / delayed[8]). Therefore, a few deepest (closest to Δ_{OUT}) key recovery rounds dominate the complexity. The enumeration of the subkey bits satisfying the transitions can be done very efficiently, since, by Table 6b, a transition gives a direct constraint on the input bit, yielding the subkey bit value or discarding the trail if the relevant input bit is constant. Pessimistically, we estimate the time complexity as 2 round decryptions per each of the final candidates, i.e., 2^{80} single-round KATAN-32 decryptions equal to $2^{73.04}$ full-round decryptions.

The final verification of the 2^{79} subkeys can be done by doing round-by-round decryption and checking conformance to the trail. Let

$$w_1, w_2, \ldots = (0,1,0,0,1,0,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1,\ldots)$$

be the weights of transitions in the 74-round trail, starting from the last round. Then, on average, we would need to decrypt each pair for

$$1 + 2^{-w_1} \cdot (1 + 2^{-w_2} \cdot (\ldots)) \leq 5.71$$

rounds, resulting in 11.42 single-round decryptions or $2^{-3.44}$ full decryptions per a full candidate subkey. The final complexity is thus $2^{75.56}$ full KATAN-32 decryptions. The total key recovery complexity is $2^{73.04} + 2^{75.56} = 2^{75.80}$ full decryptions.

[7] Experimentally, we obtained an average of $2^{33.49}$ trails which closely matches the estimates obtained by Proposition 2. See Appendix E for details.

[8] Subkey dependency matrices are provided in the supporting code repository.

Attack Complexity. The attack requires 2^{31} chosen-plaintext encryptions, $2^{33.10} \cdot 23 = 2^{37.62}$ memory blocks (for the MiF Cluster), and has time complexity of about $2^{75.80}$ full KATAN-32 decryptions. The success rate is $1 - 1/e \approx 63.2\%$ (defined solely by the main 74-round differential). It can be increased if needed by scaling the queried data.

5.4 Attack on 130-round KATAN-48

We use four 87-round differential trails with weight 46 starting with the difference $\Delta_{\text{IN}} = \texttt{000001008000}$ at the beginning of round 35. It can be verified that this difference allows 7 free rounds backward.

In KATAN-48, the two round subkey bits are used in two consecutive steps. This makes the MiF filter part weaker: rounds modeled as random suggest $2^{3.52}$ candidate trails (Proposition 2) if we directly construct the MiF filter from differences, whereas the "actual" branching factor is 2^2 simply due to 2 key bits used. This may create a problem for the gain potential, since most of the valid trails would suggest no correct subkeys (by counting reasons) and the process of discarding these trails may dominate over the subkey verification stage in terms of complexity. We resolve this problem by replacing the MiF filter with simple guessing (similar to Dinur's attack on SPECK [7]).

The attack procedure is as follows:

1. Precompute the 17.5-round forward MiF cluster from the 4 output differences Δ_{OUT} (estimated size $2^{46.93}$, see Appendix F).
2. Select 2^{44} pairs of texts with difference $\Delta_{\text{IN}} = \texttt{000001008000}$.
3. Decrypt each text for 7 rounds using zero subkeys, and query the respective ciphertext after 130 rounds (2^{45} queries).
4. For each pair of ciphertexts, decrypt 19 rounds involving 34 distinct subkey bits by recursive guessing. We expect to obtain $2^{44+34} = 2^{78}$ candidate decryptions, with estimated time complexity of about 2^{78} single-round pair decryptions ($2^{78} \cdot 2/130 = 2^{71.98}$ full-round decryptions).
5. Match the difference in the cluster to get valid trails (expecting $2^{44+34+46.93-48} = 2^{76.93}$ key-trails in total). The naive approach requires 2^{78} memory lookups in the cluster of $2^{46.93}$ trails, which can be expensive in practice. However, one can easily reduce this workload due to slow diffusion in KATAN: for example, decrypting 14 rounds (instead of 19) produces $2^{44+24} = 2^{68}$ candidates with already $48 - 5 \times 4 = 28$ bits of the final (19-round decrypted) difference available. This allows to localize memory accesses (e.g., by pooling intermediate 14-round decryptions before the other 5 rounds), dominated by 2^{78} memory accesses in clusters of size $2^{46.93-28} = 2^{18.93}$. We estimate the lookup cost to be equal to single-round decryption per pair ($2^{70.98}$ full-round decryptions).
6. Proceed with MiF round-by-round key recovery; the few first average round weights of trails as processed by MiF would be about 3.52 (see Appendix F for full trail statistics), leading to a quick reduction in the number of surviving key-trail pairs (since, again, there are only 2 subkey bits per round), before it would start increasing up to the final number.

7. By the gain theory (or weight-based calculations), we expect to arrive at $2^{-2} \cdot 2^{17 \cdot 2 + 34 + 2} = 2^{68}$ candidates for the 70 involved subkey bits after processing 17.5 rounds of MiF (the last MiF half-round uses extra 2 subkey bits).
8. Then, we continue to recursively guess subkey bits and check conformance to the trail; the last 5 rounds of the trail have weight 4, so we expect about $2^{68+5 \cdot 2 - 4} = 2^{74}$ candidates for the 80 subkey bits. Let

$$w_{82}, w_{81}, w_{80}, w_{79} \ldots = (0,0,0,1,0,0,1,0,0,1,1,0,0,0,1,0,0,0,0,1,1,0,\ldots)$$

be the weights of transitions in the 87-round trail, starting from the 82^{nd} round backward. Then, on average, we would need to decrypt each pair for

$$1 + 2^{-w_{82}} \cdot (1 + 2^{-w_{81}} \cdot (\ldots)) \leq 2^3$$

single-round decryptions of pairs, leading to complexity $2^{74} \cdot 2^3 \cdot 2/130 = 2^{70.98}$ full-round decryptions.

Attack Complexity. The attack requires 2^{45} chosen plaintext encryptions, $2^{46.24} \cdot 17 = 2^{50.33}$ memory blocks and time complexity of $2^{71.98} + 2^{70.98} + 2^{70.98} = 2^{72.98}$ full (130-round) KATAN-48 decryptions.

5.5 Attacks on 110-round KATAN-64

We use 16 differential 79-round trails with weight 60 starting with the difference $\Delta_{\text{IN}} = \texttt{0080402010000000}$ at the beginning of round 34. It can be verified that this difference allows 5 free rounds backward. Since KATAN-64 uses the two round subkey bits in three consecutive steps, we will proceed using a similar strategy to our attack on KATAN-48 to limit the branching factor to 2^2.

The attack procedure is as follows:

1. Precompute the 14-round forward MiF cluster from the 16 output differences Δ_{OUT} (estimated size $2^{60.86}$).
2. Select 2^{56} pairs of texts with difference $\Delta_{\text{IN}} = \texttt{0080402010000000}$.
3. Decrypt each text for 5 rounds using zero subkeys and query the respective ciphertext after 108 rounds (2^{57} queries).
4. For each pair of ciphertexts, decrypt 12 rounds (which includes 1 free round) involving 22 distinct subkey bits by recursive guessing (total time complexity $2^{56+22} = 2^{78}$ one-round pair decryptions or $2^{78} \cdot 2/110 = 2^{72.22}$ full-round decryptions).
5. Match the difference in the cluster to get valid trails (expecting $2^{56+22+60.86-64} = 2^{74.86}$ key-trails in total). Similarly to the attack on KATAN48, we assume the cost of 1 round decryption per lookup ($2^{71.22}$ full-round decryptions)
6. Proceed with MiF round-by-round key recovery which will quickly reduce the number of surviving key-trail pairs[9].
7. By the gain theory, we expect to have $2^{-4} 2^{14 \cdot 2 + 22} = 2^{46}$ candidates for the 2^{50} involved subkey bits.

[9] See Appendix F for full trail statistics.

8. Then, we continue to recursively guess subkey bits and check if the key-trail pairs conform to the last 15 rounds of the trail, which have weight 12. We expect around $2^{46+15\cdot 2-12} = 2^{64}$ candidates for the 80 subkey bits. Let

$$w_{64}, w_{63}, w_{62} \ldots = (1,1,1,0,1,0,0,1,0,1,0,0,1,1,1,1,2,1,1,2,2,\ldots)$$

be the weights of transitions in the 79-round trail, starting from the 64th round backward. Then, on average, we would need to decrypt each pair for

$$1 + 2^{-w_1} \cdot (1 + 2^{-w_2} \cdot (\ldots)) \leq 2^{1.21}$$

single-round decryption of pairs, leading to complexity $2^{64} \cdot 2^{1.21} \cdot 2/110 = 2^{59.43}$ full-round decryptions.

Attack Complexity. The attack requires 2^{57} chosen plaintext encryptions, $2^{60.86} \cdot 14 = 2^{64.67}$ memory blocks and time complexity is dominated by recursive guessing, which requires $2^{72.22} + 2^{71.22} = 2^{72.80}$ full KATAN-64 decryptions. Success rate is $\approx 63.2\%$.

References

1. Albrecht, M.R., Leander, G.: An all-in-one approach to differential cryptanalysis for small block ciphers. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 1–15. Springer (2012) 15, 23
2. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013) 2
3. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag, Berlin, Heidelberg (1993) 4, 6
4. Biryukov, A., dos Santos, L.C., Teh, J.S., Udovenko, A., Velichkov, V.: Meet-in-the-filter and dynamic counting with applications to speck. Cryptology ePrint Archive, Paper 2022/673 (2022), https://eprint.iacr.org/2022/673, https://eprint.iacr.org/2022/673 2, 3, 9
5. Cannière, C.D., Dunkelman, O., Knezevic, M.: KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In: CHES. Lecture Notes in Computer Science, vol. 5747, pp. 272–288. Springer (2009) 1, 2, 14
6. Chen, J., Teh, J., Liu, Z., Su, C., Samsudin, A., Xiang, Y.: Towards accurate statistical analysis of security margins: New searching strategies for differential attacks. IEEE Trans. Computers **66**(10), 1763–1777 (2017) 15, 23
7. Dinur, I.: Improved differential cryptanalysis of round-reduced Speck. In: SAC 2014. LNCS, vol. 8781, pp. 147–164. Springer (2014) 19
8. Eichlseder, M., Kales, D.: Clustering related-tweak characteristics: Application to MANTIS-6. IACR Trans. Symmetric Cryptol. **2018**(2), 111–132 (2018) 9
9. Hu, K., Cui, T., Gao, C., Wang, M.: Towards key-dependent integral and impossible differential distinguishers on 5-round AES. In: SAC. Lecture Notes in Computer Science, vol. 11349, pp. 139–162. Springer (2018) 24
10. Huang, M., Wang, L.: Automatic tool for searching for differential characteristics in ARX ciphers and applications. In: INDOCRYPT 2019. LNCS, vol. 11898, pp. 115–138. Springer (2019) 23

11. Huang, M., Wang, L.: Automatic tool for searching for differential characteristics in ARX ciphers and applications. IACR Cryptol. ePrint Arch. p. 1318 (2019) 23
12. Isobe, T., Sasaki, Y., Chen, J.: Related-key boomerang attacks on KATAN32/48/64. In: ACISP. Lecture Notes in Computer Science, vol. 7959, pp. 268–285. Springer (2013) 23
13. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of nlfsr-based cryptosystems. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 6477, pp. 130–145. Springer (2010) 15, 23
14. Knellwolf, S., Meier, W., Naya-Plasencia, M.: Conditional differential cryptanalysis of trivium and KATAN. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 200–212. Springer (2011) 23
15. Koo, B., Roh, D., Kim, H., Jung, Y., Lee, D., Kwon, D.: CHAM: A family of lightweight block ciphers for resource-constrained devices. In: ICISC. Lecture Notes in Computer Science, vol. 10779, pp. 3–25. Springer (2017) 1, 2, 10
16. Rasoolzadeh, S., Raddum, H.: Multidimensional meet in the middle cryptanalysis of KATAN. IACR Cryptol. ePrint Arch. p. 77 (2016) 14
17. Roh, D., Koo, B., Jung, Y., Jeong, I., Lee, D., Kwon, D., Kim, W.: Revised version of block cipher CHAM. In: ICISC. Lecture Notes in Computer Science, vol. 11975, pp. 1–19. Springer (2019) 1, 2, 10
18. Xing, Z., Zhang, W., Han, G.: Improved conditional differential analysis on nlfsr-based block cipher KATAN32 with MILP. Wirel. Commun. Mob. Comput. **2020**, 8883557:1–8883557:14 (2020) 23

Supplementary Material

A Comparison with Differential Attacks

To the best of our knowledge, our attack on CHAM-64 is the first third-party key recovery attack on the cipher to date.

Table 7: Summary of **differential** attacks on the KATAN family. **SK** denotes single-key and **RK** denotes related-key.

Cipher	Rounds	Type	Time	Data	Memory	Ref
KATAN-32	78	SK Conditional Differential	2^{22}	2^{16}	-	[13]
KATAN-32	98	SK Conditional Differential	2^{31}	2^{19}	-	[18]
KATAN-32	115	SK Differential	2^{79}	2^{31}	-	[1]
KATAN-32	117	SK Rectangle	$2^{79.3}$	$2^{27.3}$	$2^{29.9}$	[6]
KATAN-32	120	RK Conditional Differential	2^{31}	Practical	-	[14]
Katan-32	**123**	**SK Differential**	$\mathbf{2^{75.80}}$	$\mathbf{2^{31}}$	$\mathbf{2^{37.62}}$	**Sec 5.3**
KATAN-32	174	RK Boomerang	$2^{78.8}$	$2^{27.6}$	$2^{26.6}$	[12]
KATAN-32	187	RK Rectangle	$2^{78.4}$	$2^{31.8}$	$2^{33.9}$	[6]
KATAN-48	70	SK Conditional Differential	2^{34}	2^{31}	-	[13]
KATAN-48	87	SK Rectangle	2^{78}	$2^{36.7}$	$2^{39.3}$	[6]
KATAN-48	103	RK Conditional Differential	2^{25}	Practical	-	[14]
Katan-48	**130**	**SK Differential**	$\mathbf{2^{73.56}}$	$\mathbf{2^{45}}$	$\mathbf{2^{50.33}}$	**Sec 5.4**
KATAN-48	145	RK Boomerang	$2^{78.5}$	$2^{38.4}$	$2^{37.4}$	[12]
KATAN-48	150	RK Rectangle	$2^{77.6}$	$2^{47.2}$	$2^{49.8}$	[6]
KATAN-64	68	SK Conditional Differential	2^{35}	2^{32}	-	[13]
KATAN-64	72	SK Rectangle	2^{78}	$2^{55.1}$	$2^{58.1}$	[6]
KATAN-64	90	RK Conditional Differential	2^{27}	Practical	-	[14]
Katan-64	**109**	**SK Differential**	$\mathbf{2^{73.65}}$	$\mathbf{2^{57}}$	$\mathbf{2^{60.86}}$	**Sec 5.5**
KATAN-64	130	RK Boomerang	$2^{78.1}$	$2^{53.1}$	$2^{52.1}$	[12]
KATAN-64	133	RK Rectangle	$2^{78.5}$	$2^{58.4}$	$2^{61.4}$	[6]

B Differentials

The 38 to 42-round differentials for CHAM-64 were derived from the 39-round trail found by Huang and Wang [10, 11]. After performing a cluster search for these trails, we improved the overall differential probabilities by at least a factor of around 2^6. This significant improvement indicates that CHAM has a strong differential effect.

In the following tables, **Pr T** is the probability of the trail and **Pr D** is the probability of the differential, both expressed as $-\log_2(Pr)$. **Pr D** was computed using an SMT solver.

Table 8: CHAM-64 Differentials used in this paper.
†: the first and the last four rounds fixed to the best trail.

r	Δ_{in}	Δ_{out}	Pr T	Pr D
16	0001 8000 0080 0000	0004 0502 0088 0000	14	13.13
40	0020 0010 1020 2800	2000 1000 2810 0020	66	59.41
40	0020 0010 1020 2800	2000 1000 2810 0020†	66	60.05

Table 9: KATAN Differentials used in this paper. **Offset** denotes the starting round of the search (0 offset means the search starts from the beginning).
†: There 16 trails with the same weight starting from this difference. Only 4 are shown here.

Variant	Offset	r	Δ_{in}	Δ_{out}	Pr T	Pr D
KATAN-32	33	74	4002 8200 4002 8200	2100 0004 2100 0006	31	≥ 30.23
KATAN-48	34	87	0000 0100 8000	2000 0000 1048 2004 0000 1048 2004 0000 1049 2000 0000 1049	46	≥ 44.86
KATAN-64	33	79	0080 4020 1000 0000†	4200 1000 0010 0184 4200 1000 0010 01C0 4200 0000 0010 0184 4202 0000 0010 01C0	60	≥ 56.68

C Key-Dependency of Cham Differentials

Given a differential with probability p, one would expect to find one good pair after encrypting $\frac{1}{p}$ pairs with a success probability of around 63%. However, as shown in Table 10, this behavior significantly deviates from 16 rounds onward for CHAM. Taking the 20-round trail from the table as an example $(8004, 4082, 8200, 0100 \rightarrow 0004, 0502, 0088, 0000)$ we experimentally estimate the actual differential probability for several randomly selected master keys. The results are tabulated in Table 11. For some of these keys, the 20-round trail could in fact be impossible (key-dependent impossible differentials [9]). We leave the experimental verification of key-dependent impossibility for future work.

For keys that are valid, the actual differential probability is higher than expected. Using the same 20-round trail example, the expected differential probability after trail clustering is $2^{-19.94}$ but experimentally, we found that the actual differential probability is around $2^{-18.45}$. This weak key effect becomes more noticeable as the number of rounds increase (21 rounds: $2^{-20.24} \rightarrow 2^{-18.45}$, 22 rounds: $2^{-23.55} \rightarrow 2^{-22.7}$, 23 rounds: $2^{-24.8} \rightarrow 2^{-21.95}$, 24 rounds: $2^{-27.44} \rightarrow 2^{-24.59}$).

Table 10: CHAM-64 - Probability of finding right pairs averaged across randomly selected master keys. **Pr T** is the probability of the trail and **Pr D** is the probability of the differential, expressed as $-\log_2(Pr)$. The number of plaintext pairs used as inputs is equal to $1/PrD$. All differentials here are based on optimal trails.

Round	Pr T	Pr D	Success Prob. (%)
12	7	6.81	67.5
13	8	8	63.4
14	9	8.81	67.5
15	11	10.59	70.2
16	14	13.13	44.8
17	15	14.36	70.69
18	16	15.13	45.6
19	19	17.88	25.6
20	21	19.94	47.8
21	23	20.24	25.51
22	25	23.55	43.44
23	28	24.8	24.26
24	30	27.44	23.1

D Experimental Verification of Gain

Using CHAM as an example, we experimentally verify the time complexity estimate described in Section 3.1. To do so, we need to calculate the gain, g of the trails (or differentials) generated by MiF. For quick verification, our experiments were performed on 18 rounds, where the initial differential is over $r = 16$ rounds and there are $k = 2$ key-recovery rounds. Since the number of rounds is relatively low, Assumption 2 may not hold due to CHAM's weak diffusion. Therefore, apart from p, we will also need to compute $\tilde{p}$ to calculate g (Corollary 1). Since differentials over 18 rounds of CHAM have key dependencies (See Appendix C), both p and $\tilde{p}$ are derived experimentally for a more accurate estimate.

MiF is then used to generate a set of valid (16+2)-round trails which share the same form $\Delta_{\text{IN}} \xrightarrow{16} \Delta_{\text{OUT}} \xrightarrow{2} \Delta C$. We compute the gain for the differentials that these trails correspond to, i.e., $\Delta_{\text{IN}} \xrightarrow{16} \Delta_{\text{OUT}}$ and $\Delta_{\text{IN}} \xrightarrow{18} \Delta C$. We then perform a key-recovery attack to determine the average number of key candidates the differentials suggest, which should correspond to the gain, $g \approx 1 - \log_2(\#candidates)$. We analyze the differentials corresponding to each trail separately by fixing ΔC. To ensure that we have at least one *right* trail that follows $\Delta_{\text{IN}} \xrightarrow{16} \Delta_{\text{OUT}} \xrightarrow{2} \Delta C$, we need $1/(pq)$ pairs for the attack, where q is the differential probability of the final two (key-recovery) rounds.

Table 12 summarizes the results of this experiment for four trails that correspond to two different master keys. In all instances, the set of keys suggested by each differential includes the correct (32-bit) round keys for the final 2 rounds.

Table 11: Experimental estimates of the key-dependent differential probability for a 20-round trail ($8004, 4082, 8200, 0100 \rightarrow 0004, 0502, 0088, 0000$). All probability values are expressed as $-\log_2(Pr)$. The differential probability computed using an SMT solver was $PrD = 19.94$. **N** denotes the fact that no valid pair was found even after 2^{30} input pairs were tested.

Master Key	Pr D (Est.)
`22a9 a4b4 c8fe 4cba d139 c314 e672 9f90`	17.98
`9e96 c292 ca49 4101 97aa a4b9 7cf6 e794`	17.45
`214b 2fd7 16a9 9cd6 d003 dba8 0c87 835e`	18.00
`4661 480e 53e9 5fda 0f46 6169 3044 d509`	N
`23a8 d74f 7698 65ef 11a4 7198 6ff1 0c34`	N

In all four examples, the actual gain generally follows the estimated gain albeit with some negligible error of around 1 bit.

Table 12: Experimental verification of the gain, g of differentials produced by MiF performed on 18 rounds of CHAM-64. All trails have Δ_{IN} = `0001 8000 0080 0000` and Δ_{OUT} = `0004 0502 0088 0000`. All probability values are expressed as $-\log_2(Pr)$.
Key 1 = `58ec 944a 5cff dc51 4873 9869 23c6 4567`
Key 2 = `fc99 9a30 cbf0 776c 7008 9a6a a7dd dd1b`

Key	ΔC	p	$\tilde{p}$	q	Est. Gain	Actual Gain
1	`0088 0000 001e 3a05`	11.87	18.2	10	6.33	7.1
1	`0088 0000 0006 7a05`	11.87	18.95	10	7.08	7.26
2	`0088 0000 001a e604`	12.68	20.86	12	8.18	7.07
2	`0088 0000 0036 1a1d`	12.68	21.1	12	8.42	7.93

E Experimental Verification of the KATAN-32 Attack

In this section, we experimentally verify the trail statistics involved in the 124-round attack on KATAN-32 (Section 5.3). We also provide some insights into the actual trails that were generated by MiF.

Recall that the MiF procedure was performed over 42 rounds, with the first 23 rounds propagating forward from two differences Δ_{OUT}, and the final 19 rounds propagating backward from (partially decrypted) ciphertext differences ΔC. In our experiments, we first constructed the MiF cluster that consists of $2^{30.36} + 2^{32.87} = 2^{33.10}$ 23-round trails. We computed both the actual and estimated (using Assumption 3) average trail weights for all 23 rounds and found that the values match. The average trail weights per round are listed in Table 13.

Table 13: Average trail weights for the first 23 MiF rounds from the 124-round attack on KATAN-32, starting from difference Δ_{OUT}. Since the actual and estimated average trail weights (based on Assumption 3) are the same up to 2 weight digits, we only list the values once in the table.

Δ_{OUT} = (2100 0004)				Δ_{OUT} = (2100 0006)			
Round	Weight	Round	Weight	Round	Weight	Round	Weight
108	1	120	1.44	108	1	120	1.39
109	1	121	1.62	109	1	121	1.6
110	0	122	1.97	110	1	122	1.96
111	1	123	1.46	111	1	123	1.76
112	0.58	124	1.78	112	0.58	124	1.84
113	0	125	1.65	113	0	125	1.76
114	1	126	1.8	114	1	126	1.8
115	1.17	127	1.78	115	1.59	127	1.83
116	1	128	1.79	116	1.58	128	1.78
117	0.92	129	1.82	117	1	129	1.78
118	1.87	130	1.81	118	1.83	130	1.8
119	1.9			119	2		

We then encrypted pairs of random plaintexts with difference Δ_{IN} over 115 rounds starting from round 34 to obtain ciphertext pairs with difference ΔC. We implicitly assume that these plaintexts (and their resulting ciphertext) differences will be additionally decrypted (and resp. encrypted) using zero subkey bits over the 4 free rounds.

For each of the ciphertext differences, we enumerate all possible trails backward in search of a match in the cluster. Based on Proposition 2, we estimate that there will be $2^{33.44}$ trails on average for each difference. From 10 randomly sampled ciphertext pairs, the actual number of trails on average was $2^{33.49}$, which closely approximates our earlier estimate. For 2^{30} pairs, we would then have a MiF filter size of approximately $2^{63.49}$ trails.

On average, each ciphertext difference leads to $2^{33.49+33.10-32} = 2^{34.59}$ trails that match the MiF cluster. For 2^{30} pairs, we would then expect around $2^{64.59}$ trails suggested by MiF. Since the ciphertext difference is random-like, extending it backward should lead to around $2^{1.76}$ trails per round (by Proposition 2). Based on Assumption 3, the average trail weights are estimated to be around 1.76. This is a close approximation of the computed actual average trail weights for these trails as summarized in Table 14.

Table 14: Average round trail weights for the 18/19 bottom MiF rounds from the 124-round attack on KATAN-32. **All Trails** refers to the entire set of trails in the MiF filter while **MiF Trails** are trails from the MiF filter that found a match in the cluster.

Round	All Trails	MiF Trails	Round	All Trails	MiF Trails
131	1.80	1.80	140	1.72	1.87
132	1.80	1.82	141	1.77	1.85
133	1.80	1.80	142	1.53	1.86
134	1.76	1.83	143	1.74	1.85
135	1.79	1.83	144	1.32	1.95
136	1.80	1.82	145	1.23	1.41
137	1.73	1.81	146	1.51	1.55
138	1.87	1.77	147	1.74	1.90
139	1.67	1.89	148	1.23	1.83

F KATAN Trail Distributions

In the following tables, we report the trail statistics from a set of differences Δ_{OUT} for KATAN-48 and KATAN-64. For KATAN-48, there are 4 differences in the set while for KATAN-64, there are 16. The full trails in these sets are available in our GitHub repository. All differences in these sets form truncated differences. For KATAN-48 the truncated difference (in binary) is

```
00100000 00000*00 00000000 00000000 00010000 0100100*
```

whereas for KATAN-64, it is

```
01000010 000000*0 000*0000 00000000 00000000 00010000
00000001 1*000*00 .
```

Table 15: Trail extensions for the KATAN family starting from the differences used in the attacks.
†: Estimated number of trails based on Proposition 2.

(a) KATAN-48 trail statistics from 4 differences

Round	#Trails	Avg. Weight (this round)
1	2^{4}	4
2	$2^{5.58}$	1.58
3	$2^{5.58}$	0
4	$2^{7.17}$	1.58
5	$2^{9.34}$	2.17
6	$2^{11.66}$	2.32
7	$2^{13.40}$	1.74
8	$2^{16.32}$	2.92
9	$2^{19.23}$	2.91
10	$2^{21.55}$	2.32
11	$2^{24.39}$	2.84
12	$2^{27.87}$	3.47
13	$2^{30.93}$	3.07
14	$2^{34.54}$	3.61
15	$2^{38.06}$	3.52
16	$2^{41.59}$	3.53
17	$2^{45.17}$	3.58
17.5	$2^{46.93}$	1.76

(b) KATAN-64 trail statistics from 16 differences.

Round	#Trails	Avg. Weight (this round)
1	$2^{7.58}$	7.58
2	$2^{8.17}$	0.58
3	$2^{11.49}$	3.32
4	$2^{12.91}$	1.42
5	$2^{16.88}$	3.98
6	$2^{21.28}$	4.40
7	$2^{24.82}$	3.54
8	$2^{29.07}$	4.25
9	$2^{34.34}$	5.27
10	$2^{39.74}$	5.40
11	$2^{45.02}$ †	5.28
12	$2^{50.30}$ †	5.28
13	$2^{55.58}$ †	5.28
14	$2^{60.96}$ †	5.28