Cryptanalysis of Full SCARF

Antonio Flórez-Gutiérrez[1], Eran Lambooij[2], Gaëtan Leurent[3], Håvard Raddum[4], Tyge Tiessen[5], and Michiel Verbauwhede[6]

[1] NTT Social Informatics Laboratories, NTT Corporation, Japan
antonio.florez@ntt.com

[2] Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
eran.lambooij@biu.ac.il

[3] Inria, Paris, France gaetan.leurent@inria.fr

[4] Simula UiB, Bergen, Norway haavardr@simula.no

[5] Technical University of Denmark, Denmark tyti@dtu.dk

[6] COSIC, KU Leuven, Leuven, Belgium michiel.verbauwhede@esat.kuleuven.be

Abstract. SCARF is a tweakable block cipher dedicated to cache address randomization, proposed at the USENIX Security conference. It has a 10-bit block, 48-bit tweak, and 240-bit key. SCARF is aggressively optimized to meet the harsh latency constraints of cache address randomization, and uses a dedicated model for its security claim.
The full version of SCARF has 8 rounds, and its designers claim security up to 2^{40} queries and 2^{80} computations. In this work we present a distinguisher against 6-round SCARF under the collision model with time and query complexity 2^{30}, and a key-recovery attack against the full 8-round SCARF under the encryption-decryption model with 2^{39} queries and time $2^{76.2}$. As part of the attack, we present a novel method to compute the minimal number of right pairs following a differential characteristic when the input pairs are restricted to a subspace of the domain of the primitive.

Keywords: SCARF · Tweakable block cipher · Cryptanalysis

1 Introduction

Block ciphers are among the most commonly used and versatile primitives in symmetric cryptography, and are essential to the encryption of sensitive data in countless applications. More generally, block ciphers are useful in scenarios beyond encryption or authentication. One such case is that of cache randomization.

Cache randomization [14] aims to prevent an adversary from gaining sensitive information by timing memory accesses (whose latency greatly depends on whether the associated address was stored in cache or not) on certain CPU architectures. This is achieved by obfuscating the relationship between memory addresses and the cache registers assigned to them. In recent cache randomization schemes [12,15,13], block ciphers have been used to provide a secure source of randomization while allowing for regular remapping via key rotations. As these

Table 1: Attacks on SCARF

Rounds	Model	Queries	Time	Note	Reference
4	enc-dec	2^{10}	2^{61}	Key recovery	[9]
7	enc-dec	2^{40}	2^{76}	Key recovery	[7]
8	enc-dec	2^{63}	2^{68}	Multikey distinguisher	[7]
6	collision	2^{30}	2^{30}	Distinguisher	Section 4
8	enc-dec	2^{39}	2^{77}	Key recovery	Section 5

block ciphers are used as part of the performance critical lookup path, low latency is an essential requirement.

SCARF [8] is a novel tweakable block cipher designed with the specific needs of cache randomization in mind. It uses the index and the tag sections of a memory address as inputs. The tag is used as the plaintext and the index as a tweak. The output (ciphertext) is then used to determine which cache set the register will be stored in. This randomisation is performed under a secret key, making it difficult for an attacker to manipulate the cache in a way which can intentionally interfere with the memory space of other processes.

The exact specifications of SCARF are somewhat unusual. SCARF features a 240-bit key, the index used as tweak has 48 bits, while the tag used as the plaintext/ciphertext is only 10 bits long. The SCARF data encryption path consists of eight rounds which manipulate the 10-bit state, and the tweakey schedule generates the round subkeys from the 48-bit tweak and the 240-bit key. One of the features of SCARF's design is that 30 bits of key material are absorbed into the 10-bit state at each round.

Another feature of the design is the security model. Instead of aiming for the traditional PRP security which is expected from most block ciphers, SCARF's creators propose two weaker security models where ciphertexts cannot be observed directly. In the first model, attackers have access to a collision oracle: querying two plaintext-tweak pairs $(P_1, T_1), (P_2, T_2)$, they learn whether the corresponding ciphertexts collide $(E_{K,T_1}(P_1) = E_{K,T_2}(P_2))$. This model corresponds to the cache randomization scenario, where an adversary can only detect addresses which map to the same cache register. In the second model, attackers have access to an encrypt-then-decrypt oracle: given a plaintext P and two tweaks T_1, T_2, they receive $E^{-1}_{K,T_2}\big(E_{K,T_1}(P)\big)$. The second model gives more power to the adversary, and is more readily compatible with existing cryptanalysis techniques.

The authors argue that in these security models an adversary would essentially need to attack SCARF with double the number of rounds. For this reason, only eight rounds are used, which greatly lowers the latency of a SCARF encryption. The designers claim that, under both security models, SCARF cannot be broken with less than 2^{40} oracle queries and less than 2^{80} time.

Previous Works. Given SCARF's recent introduction, there are few published results on third-party cryptanalysis. To the best of our knowledge, there are only two relevant publications: the works of Chen et al. [9] and of Boura et al. [7]. Chen et al. [9] present a meet-in-the-middle attack on SCARF reduced to four rounds with a time complexity of $2^{60.63}$ encryptions and query complexity 2^{10} using the encryption-decryption oracle. This attack is also applicable with the collision oracle, though with query complexity raised to 2^{20}.

Boura et al. target the encryption-decryption model, and thus analyze differentials for $R + R$ rounds of SCARF. These differentials are used in a key recovery attack on 7-round SCARF with (maximum) 2^{40} query complexity and 2^{76} time complexity. A distinguishing attack on 8+8 rounds in the multi-key setting with query complexity 2^{63} and time complexity 2^{68} is also shown.

Our Results. This paper provides the first cryptanalysis on full 8-round SCARF whose complexity lies within the security claims of the designers. It is a key recovery attack using techniques from differential cryptanalysis. We first propose an attack on 6-round SCARF as a stepping stone towards the attack on the full cipher. The main attack is split into three stages. In the first stage we identify 2^{36} candidates for a 70-bit segment of the 240-bit key, containing the correct key. This is the most costly step of the attack and has data complexity 2^{39} and time complexity $2^{76.2}$. For this stage we also develop a novel method for computing a tight lower bound on the number of right pairs over all guessed key. In the second stage we uniquely determine the correct value of the 70-bit part of the key. The third stage of the attack recovers another 60 bits from the key. After the third stage the remaining cipher is reduced to just four rounds which still contain unrecovered key bits. It should then be easy to recover the remaining unknown key bits, although we do not explicitly give an algorithm for it. In Table 1 we summarize our results and compare them to the two previous analyses of SCARF.

Outline. This paper is organized as follows. In Section 2 we describe SCARF and introduce notation, and in Section 3 we revisit the particular security model for SCARF. Section 4 introduces a distinguisher for 6-round SCARF which will be used to stage the attack on full SCARF given in Section 5 and Section 6, before concluding in Section 7.

2 Preliminaries

2.1 Notation

Table 2 shows the most important notational conventions used in this paper. We use $\oplus$ for both the exclusive or of bits or binary vectors and the direct sum of vector spaces, but the meaning should be clear from context. Following the SCARF specification, round and subkeys are indexed starting from 1. We also number the S-boxes in the SL layer starting from 1, bits are indexed starting from 0, where the least significant bit occupies the rightmost position 0.

Table 2: Notation used in the paper

P	10-bit SCARF plaintext
T	48-bit tweak used in SCARF tweakey schedule
K	240-bit user-selected key
K^i	60-bit part of K entered into tweakey schedule
$k^{(j)}$	j-th round key
$k_i^{(j)}$	5-bit subkey i used in round j
$X[i]$	bit i in the bit-string X
$X[i:j]$	the bit-string $(X[i], \ldots, X[j])$
`1f`	teletype typeface is used for hexadecimal values
`0b11111`	binary values are prefixed with `0b`

2.2 SCARF

SCARF was introduced at the USENIX Security Conference [8], and is a tweakable block cipher designed specifically for the purpose of cache randomization. It has a 10-bit block length, and takes a 48-bit tweak and a 240-bit key.

Specification. SCARF consists of a data encryption path and a tweakey schedule (see Figure 1). The tweakey schedule takes the 48-bit tweak T and the 240-bit key K and outputs eight 30-bit round subkeys $k^{(i)}$. The data encryption path takes the 10-bit plaintext P and the subkeys, and outputs the ciphertext C.

Data Encryption Path. Encryption consists of eight consecutive rounds, the first seven of which are identical and denoted R_1, and the last of which is denoted R_2. In both cases, the 10-bit input X is split into two 5-bit parts $X_L \| X_R$. Similarly, the 30-bit round subkey k is split into six five-bit parts $k_6\|k_5\|k_4\|k_3\|k_2\|k_1$. The standard round function R_1 operates as follows:

$$\begin{aligned} Y &\leftarrow G(X_L, k_1, k_2, k_3, k_4, k_5) \oplus X_R; \\ X_R &\leftarrow S(X_L \oplus k_6); \\ X_L &\leftarrow Y; \end{aligned}$$

where (see Figure 2)

$$\begin{aligned} G(x, k_1, k_2, k_3, k_4, k_5) &= \left[\bigoplus_{j=0}^{4} ((x \lll j) \wedge k_{j+1}) \right] \oplus ((x \lll 1) \wedge (x \lll 2)), \\ &= M \cdot x \oplus g(x), \end{aligned}$$

where M is a key-dependent matrix whose elements are $m_{ij} = (k_{(j-i \bmod 5)+1})_i$ and $g(x) = (x \lll 1) \wedge (x \ll 2)$ is the nonlinear part.

The Sbox S is defined as follows:

$$\begin{aligned} S(x) = &\Big((x \vee (x \lll 1)) \wedge \Big(\overline{(x \lll 3)} \vee \overline{(x \lll 4)} \Big) \Big) \\ &\oplus \Big((x \vee (x \lll 2)) \wedge \Big(\overline{(x \lll 2)} \vee (x \lll 3) \Big) \Big). \end{aligned}$$

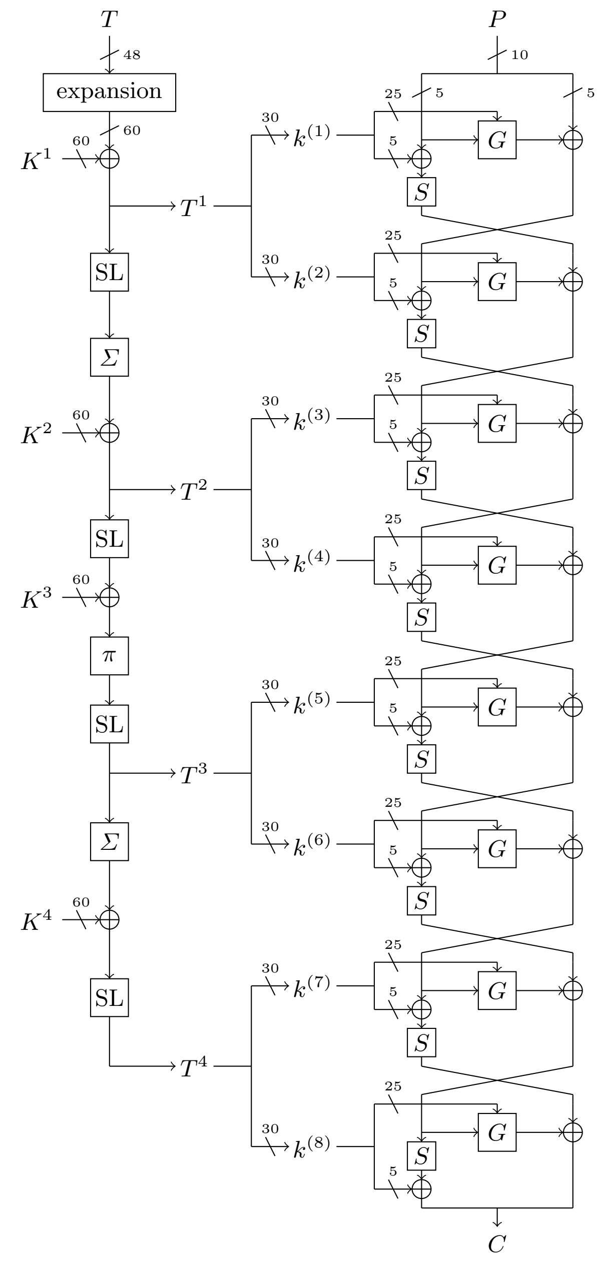


Fig. 1: The SCARF cipher, including the tweakey schedule.

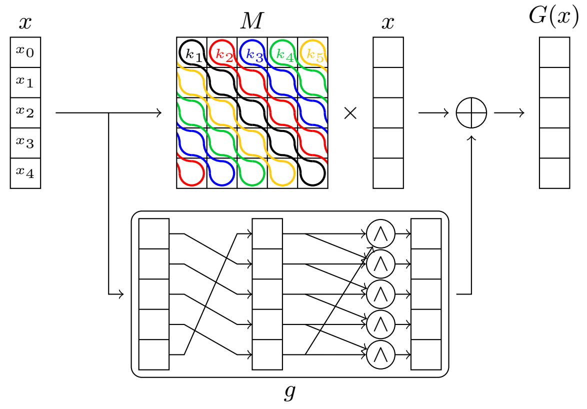


Fig. 2: The keyed nonlinear function G.

The last round function R_2 operates as follows:

$$x_R \leftarrow G(x_L, k_1, k_2, k_3, k_4, k_5) \oplus x_R;$$
$$x_L \leftarrow S(x_L) \oplus k_6;$$

Tweakey Schedule. The tweakey schedule takes the 48-bit tweak T and the key K, which is divided into four 60-bit parts $K^4 \| K^3 \| K^2 \| K^1$, and generates the eight 30-bit subkeys as follows:

$$\begin{aligned}(k^{(2)} \| k^{(1)}) &= T^1 \leftarrow \text{expansion}(T) \oplus K^1;\\ (k^{(4)} \| k^{(3)}) &= T^2 \leftarrow \Sigma(\text{SL}(T^1)) \oplus K^2;\\ (k^{(6)} \| k^{(5)}) &= T^3 \leftarrow \text{SL}(\pi(\text{SL}(T^2) \oplus K^3));\\ (k^{(8)} \| k^{(7)}) &= T^4 \leftarrow \text{SL}(\Sigma(T^3) \oplus K^4);\end{aligned}$$

where

$$\begin{aligned}\text{expansion}(T) &= (0 \| T[47:44] \| 0 \| T[43:40] \| 0 \| \cdots \| 0 \| T[3:0]),\\ \Sigma(x) &= x \oplus (x \lll 6) \oplus (x \lll 12) \oplus (x \lll 19)\\ &\quad \oplus (x \lll 29) \oplus (x \lll 43) \oplus (x \lll 51),\end{aligned}$$

SL is the parallel application of 12 5-bit Sboxes S (the same Sbox as in the data encryption path), and π is a bit permutation which takes x_i to x_{p_i} where $p_i = 5i \bmod 59$ if $i \neq 59$ and $p_{59} = 59$.

Security Claim. Because of the extremely tight latency restrictions which are imposed by cache randomisation, the designers do not target PRP security, as is usual for block ciphers. Instead, they note that, since the attacker cannot directly observe which cache set a particular memory address is mapped to, and can instead only detect addresses which map to the same (unknown) set, they can only access a *collision oracle* that returns one if and only if, under the same key and two different tweaks, the encryptions of two plaintexts collide, *i.e.*, $E_{K,T_1}(P_1) = E_{K,T_2}(P_2)$. The designers limit the amount of queries to this oracle to 2^{40} and the computational complexity of the attack to 2^{80}. We note that there is no known attack matching this bound, it is just a limited claim due to the trade-offs between efficiency and security.

Since this model is somewhat unusual when trying to apply existing cryptanalysis tools, the designers propose an additional oracle, the *encryption-decryption oracle*. In this oracle the attacker can request that a plaintext is first encrypted under some tweak T_1 and then decrypted under another tweak T_2 (under the same unknown key), *i.e.*, $E^{-1}_{K,T_2}(E_{K,T_1}(P))$. As with the collision model the adversary can do at most 2^{40} queries and 2^{80} computations.

In Section 3 we give a formal definition of these models and discuss how they can be leveraged for cryptanalysis.

3 SCARF Security Model

First we look at the formal definitions of the security models given in [8] for SCARF. The designers define two models: the collision model, where the attacker can observe if two encryptions collide, and the stronger encryption-decryption model, where the attacker can observe the output after sequentially encrypting and decrypting a plaintext under two tweaks.

Security Requirement 1 (Collision model[8]) *Let $\mathcal{O}_{real}$ be an oracle which, given a pair of memory addresses (P_1, T_1), (P_2, T_2), returns whether $E_{K,T_1}(P_1) = E_{K,T_2}(P_2)$, where E is an instance of SCARF for a secret key K. Let $\mathcal{O}_{ideal}$ be a similar oracle but for which E is a tweakable random permutation Π. If restricted to 2^{40} queries to the oracle and 2^{80} time, an attacker cannot distinguish between $\mathcal{O}_{real}$ and $\mathcal{O}_{ideal}$.*

In other words, the designers assume that the attacker cannot observe the ciphertexts and can instead only query pairs of inputs to determine whether they lead to a collision or not. Since this security requirement is difficult to manipulate for cryptanalysis, the following stronger security claim is also used:

Security Requirement 2 (Encryption-decryption model[8]) *Let $\mathcal{O}_{real}$ be an oracle which, given a plaintext P_1 and a pair of tweaks T_1, T_2, returns $P_2 = E^{-1}_{K,T_2}(E_{K,T_1}(P_1))$, where E is an instance of SCARF for a secret key K. Let $\mathcal{O}_{ideal}$ be a similar oracle but for which E is a random tweakable permutation Π. If restricted to 2^{40} queries to the oracle and 2^{80} time, an attacker cannot distinguish between $\mathcal{O}_{real}$ and $\mathcal{O}_{ideal}$.*

In summary, the attacker is able to query the composition of encryption and decryption under two different tweaks. The designers note that, in practice, the encryption-decryption model corresponds to SCARF encryption under double the number of rounds. This means they can use just 8 rounds, leading to an extremely competitive latency, and claim that an attacker must deal with a much more robust 16-round cipher.

3.1 Relations Between the Models

Two important observations are discussed in [8, Appendix A.1]. First, one can switch between the encryption-decryption model and the collision model. For a given pair of tweaks (T_1, T_2), an attacker can reconstruct the full encryption-decryption codebook (corresponding to 2^{10} data) using about 2^{18} queries to the collision oracle. Reciprocally, given access to the encryption-decryption oracle, the attacker can query the full codebook with $2^{10} - 1$ queries[7], and emulate any collision query with tweaks (T_1, T_2).

Second, given the full codebooks of the encryption-decryption oracle with tweaks (T_1, T_2) and (T_2, T_3), we can combine the data to reconstruct the encryption-decryption codebooks with tweaks (T_1, T_3):

$$E^{-1}_{K,T_3}(E_{K,T_1}(P)) = E^{-1}_{K,T_3}\left(E_{K,T_2}\left(E^{-1}_{K,T_2}(E_{K,T_1}(P))\right)\right)$$

In particular, if we query the full codebook with tweaks (T_i, T^*) for a fixed tweak T^* and $T_i \in \mathcal{T}$, we deduce the full codebook with any pair of tweaks $(T_i, T_j) \in \mathcal{T} \times \mathcal{T}$.

3.2 Attacks Based on Collisions

Our attacks are based on differential cryptanalysis: we build pairs of tweak and plaintext (P_1, T_1) and (P_2, T_2) trying to control the internal differences in the key schedule and data path. However, we focus on specific characteristics leading to ciphertext collisions. Therefore, we can directly use the collision oracle, and we only have to study 8 rounds of the cipher rather than 16.

When given access to the encryption-decryption oracle, we use it with a fixed tweak T^* for decryption. We model this oracle as $\Pi(E_{K,T}(P))$ with Π a fixed but unknown permutation ($\Pi = E^{-1}_{K,T^*}$). After querying $\Pi(E_{K,T}(P))$ for the full codebook under a fixed set $\mathcal{T}$ of tweaks, we can check for collisions $E_{K,T_1}(P_1) = E_{K,T_2}(P_2)$ for any $T_1, T_2 \in \mathcal{T}$ and any $P_1, P_2, \in \{0,1\}^{10}$:

$$E_{K,T_1}(P_1) = E_{K,T_2}(P_2) \iff \Pi(E_{K,T_1}(P_1)) = \Pi(E_{K,T_2}(P_2))$$

With this approach, we have more flexibility in our queries: we have the ability to use structures of tweaks, decreasing the complexity of some attacks. Moreover, we still consider only 8 rounds in the analysis.

[7] Actually, $2^{10} - 2$ queries are sufficient because the permutation is even.

4 Distinguishing 6-round SCARF

In this section, we present a distinguisher against six rounds of SCARF. This distinguisher is a stepping stone to the more elaborate eight-round attack described in Section 5. It is furthermore of interest since it is an efficient distinguisher in the *collision model*.

The main idea of the distinguisher is to look for a pair of tweaks together with a plaintext such that the ciphertexts corresponding to the encryption of the plaintext under the two tweaks have an unusually high collision probability.

In our distinguisher, we fix the plaintext to the all-zeroes plaintext, and attempt to choose the tweaks so that the data paths collide on all intermediate states with high probability. As we will show later, this probability turns out to be particularly high if we choose a tweak pair with a difference of `8`.

First, we look at the probability that a difference in the round subkeys introduces a difference in the state when applying the G function. Next, we analyze the probability that data paths collide under different round subkeys. Finally, we discuss the probability that an initial tweak difference generates conforming subkey differences when it propagates through the tweakkey schedule.

4.1 Analysis of the G Function

The function G is the combination of a left-multiplication by a key-dependent matrix M and a fixed non-linear function g:

$$G(x) = M \times x \oplus g(x)$$

Two different round keys give two different matrices M and M' and thus two different functions G and G' for which the following holds:

$$G(x) = G'(x) \iff x \in \ker(M \oplus M').$$

It follows that the probability that G and G' collide for a random x is $2^{-\operatorname{rank}(M \oplus M')}$. For two uniformly random matrices M, M' the expected probability is $2^{-4.023}$.

4.2 Data Path Analysis

The data path used in the distinguisher is depicted in Figure 3. The first thing to note is that we only consider round key differences that are always zero in the $k_6^{(i)}$ part of each round key. The only way to introduce a difference into the state is thus via the G function. Let us now look at the probability that this happens for the specific round key differences depicted in Figure 3. We will look at the probability that these differences occur later.

1. We start with the all-zeroes plaintext, so the G function cannot introduce a difference in the first round since $G(0) = 0$ for any round key.
2. In the second round, no state difference is introduced as there is no difference in the round keys.

- ☐ Inactive key bit
- ▨ Potentially active key bit
- 🟥 Active key bit
- ↘ Inactive word
- ↘ Active word
- ⇣ Computation direction

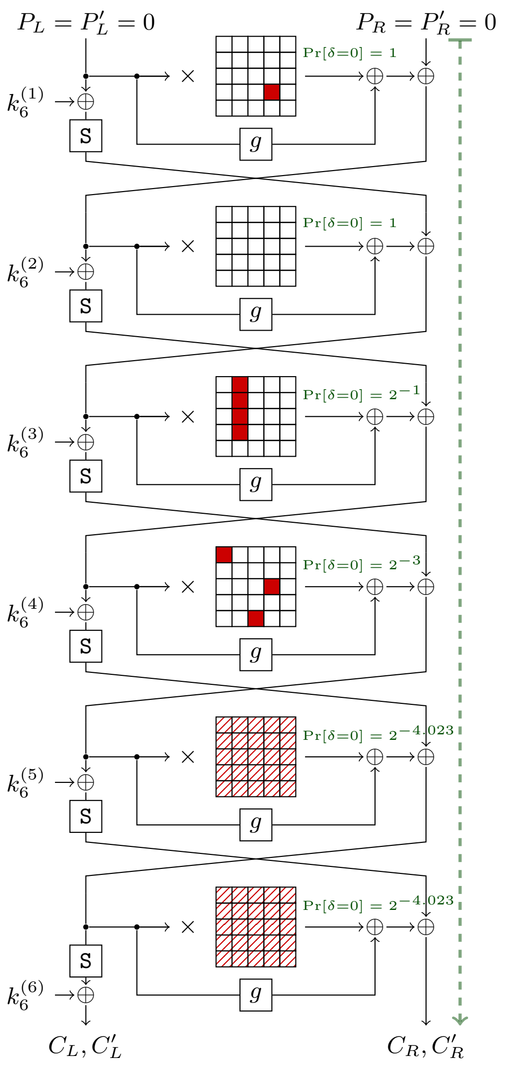


Fig. 3: Data trail for the 6-round attack.

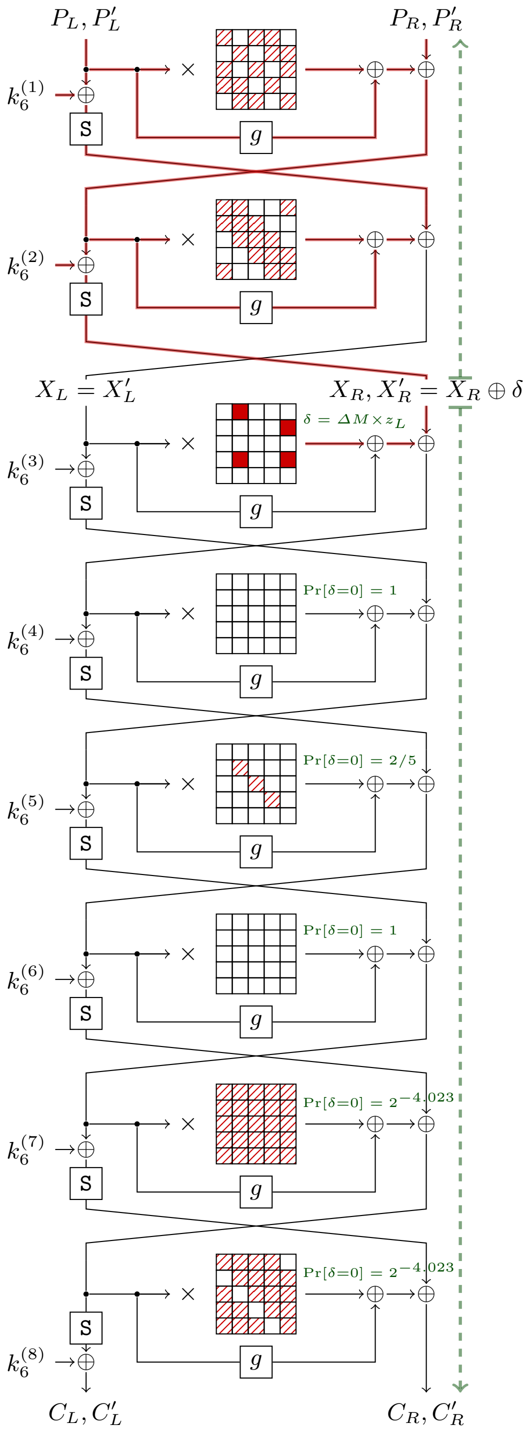


Fig. 4: Data trail for the 8-round attack.

3. In the third round we have 4 active bits in M, but the matrix $M \oplus M'$ only has rank 1. Therefore, the collision is preserved with probability 2^{-1}.
4. In the fourth round there are three active bits in M, and the matrix $M \oplus M'$ has rank 3. Therefore, the collision is preserved with probability 2^{-3}.
5. In the fifth and sixth round, we consider essentially random matrices M and M'. The probability that a vector lies in the kernel of a random matrix of this size is $2^{-4.023}$.

In summary, assuming a key/tweak pair which follows the given subkey differences, we estimate that the all-zeroes plaintext will collide at every round with probability $2^{-12.05}$.

4.3 Tweakey Schedule Analysis

As explained earlier, we want the differences in the k_6-parts of the round subkeys to be zero and the probability that G preserves collisions to be high at each round. By examining the tweakey schedule we can pick an input difference such that the probability of the $k_6^{(i)}$ having zero differences is as high as possible, while the kernel of the matrices $M \oplus M'$ is as large as possible. This will maximize the probability of getting a collision after six rounds.

The related-tweak characteristic that we consider is shown in Figure 5, and starts with the tweak difference `8`. Since the tweak and key are merged linearly, the differences in the first two round subkeys are known:

$$k^{(1)} \oplus k'^{(1)} = (\mathtt{8}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}) \quad \text{and} \quad k^{(2)} \oplus k'^{(2)} = (\mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0})$$

With probability 2^{-3} the active S-box in the first S-box layer of the tweakey schedule follows the transition $\mathtt{8} \to \mathtt{2}$. After applying Σ and the bit permutation π, we get the following key state differences:

$$k^{(3)} \oplus k'^{(3)} = (\mathtt{2}, \mathtt{4}, \mathtt{8}, \mathtt{0}, \mathtt{1}, \mathtt{0}) \quad \text{and} \quad k^{(4)} \oplus k'^{(4)} = (\mathtt{1}, \mathtt{0}, \mathtt{10}, \mathtt{0}, \mathtt{4}, \mathtt{0})$$

In rounds five and six of the tweakey schedule, we are only interested in the probability that there is no difference in the k_6 part of the round keys (*i.e.*, $k_6^{(5)}$, and $k_6^{(6)}$). If we trace the bits that lead to $k_6^{(5)}$ and $k_6^{(6)}$ in T^3, we can see that this event happens with probability $2^{-4.68}$, since 5 bits are already inactive, 4 bits are inactive with probability 1/2 and one bit is inactive with probability 5/8.

$$k^{(5)} \oplus k'^{(5)} = (*, *, *, *, *, \mathtt{0}) \quad \text{and} \quad k^{(6)} \oplus k'^{(6)} = (*, *, *, *, *, \mathtt{0})$$

Therefore, a random key/tweak pair follows this trail with probability $2^{-7.68}$, resulting in zero differences in $k_6^{(1)}$, $k_6^{(2)}$, $k_6^{(3)}$, $k_6^{(4)}$, $k_6^{(5)}$, $k_6^{(6)}$; and difference matrices $M \oplus M'$ of ranks 0, 1, and 3 in rounds 2, 3, and 4, respectively.

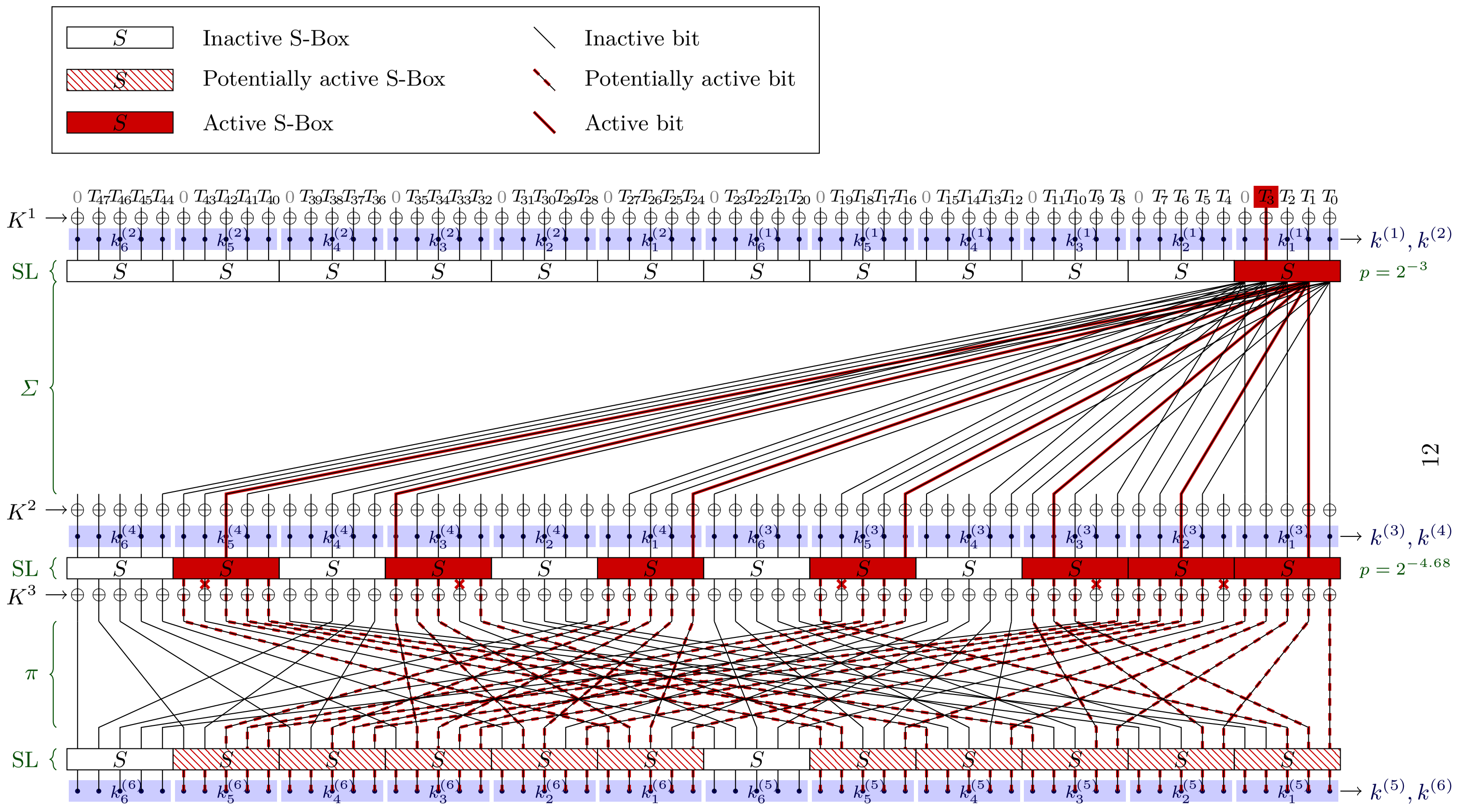


Fig. 5: Related-tweak trail in the key-schedule of SCARF for the 6-round attack.

4.4 The Full Distinguisher

We now know that the probability that a tweak pair with starting difference $\mathtt{8}$ follows our trail is $2^{-7.68}$. We also established in Section 4.2 that the probability that the all-zeroes plaintext leads to a ciphertext collision under a tweak pair which follows the trail is $2^{-12.05}$. We conclude that the probability that we obtain a pair of tweaks and plaintexts following the trail is $2^{-7.68} \times 2^{-12.05} = 2^{-19.73}$. We assume that the probability of collision when not following the trail is 2^{-10}, so we expect a total collision probability of $2^{-19.73} + \left(1 - 2^{-19.73}\right) \times 2^{-10} \approx 2^{-10} + 2^{-19.73}$ for pairs $(0, T), (0, T + \mathtt{8})$, over random keys and random tweaks.

Experimentally we observe a higher probability than explained by this trail alone. For a random key K and a random tweak T, we observe:

$$\Pr_{K,T}\left[E_{K,T}(0) = E_{K,T+\mathtt{8}}(0)\right] \approx 1.004 \times 2^{-10} \approx 2^{-10} + 2^{-17.8}$$

However, the probability is not uniform over all keys: some keys lead to a higher bias than others under random tweaks. Out of 64 random keys, with 100×2^{30} samples for each key, we observe probabilities of collision between $1.0024 \times 2^{-10} \approx 2^{-10} + 2^{-18.7}$ and $1.0069 \times 2^{-10} \approx 2^{-10} + 2^{-17.2}$. In particular we observe four clusters of keys of similar size, with collision probabilities respectively around 1.0025×2^{-10}, 1.0037×2^{-10}, 1.0052×2^{-10}, and 1.0065×2^{-10}.

We tried to find alternative trails explaining the higher bias, but did not identify any. Since the distinguisher is already practical, we did not pursue this analysis further.

4.5 Using the Distinguisher

In order to distinguish 6-round SCARF from a random tweakable permutation in the collision model, we query the oracle with n pairs $(0, T), (0, T + \mathtt{8})$ for random tweaks T and count the number of corresponding collisions.

If the oracle is a random tweakable permutation, each query returns independent results, and a collision occurs with probability $q = 2^{-10}$. Therefore, the number of collisions follows a binomial distribution $B(n, q)$ with n trials, and probability q. If the oracle is 6-round SCARF, we expect that each query returns a collision with probability $p = 2^{-10} + 2^{-17.8}$, so the number of collisions follows a binomial distribution $B(n, p)$.

We can distinguish both cases by counting the number of collisions, and comparing the relative frequency of collisions to a threshold $t = (p + q)/2$. Given enough samples, the observed probability is likely to be above the threshold for 6-round SCARF, and below the threshold for a random tweakable permutation. Since we observed a non-uniform collision probability over the keyspace, we set the threshold based on the cluster of keys with the lowest collision probability $p \approx 1.0025 \times 2^{-10} \approx 2^{-10} + 2^{-18.6}$: we use $t = (p+q)/2 = (1+1.0025)/2 \times 2^{-10} = 1.0013 \times 2^{-10}$. Following the analysis of [10, Theorem 2], the number of samples required to distinguish the two distributions is in the order of $\mathcal{O}\left(p/(q-p)^2\right)$. Therefore we need $c \times 2^{2\times 18.6-10} = c \times 2^{27.2}$ samples, with c a small constant depending on the required success rate. More precisely, we can estimate the error rate using a Poisson approximation or a normal approximation of the binomial distribution. With $n = 2^{30}$ samples, we expect an error rate of 0.11.

Experimentally, we have simulated the distinguisher with $n = 2^{30}$ pairs of encryption. We use 6-round SCARF with 64 random keys on the one hand, and 64 sets of random ciphertextext on the other hand, running 100 experiments in each case (with different subsets of the tweak space). We observe between 2% and 16% of false positive, and between 0% and 16% of false negative (in particular, 34 keys result in 100 successes because they correspond to a higher collision probability). This matches the theoretical analysis.

5 Breaking 8-round SCARF

In order to attack 8-round SCARF, our strategy is to guess keys in the first rounds, and use a distinguisher based on a collision characteristic (similar to the 6-round distinguisher) for the later rounds. In particular, by guessing 60 bits of the master key (corresponding to K^1), we can compute the subkeys of the first two rounds $(k^{(1)}, k^{(2)})$ for any tweak.

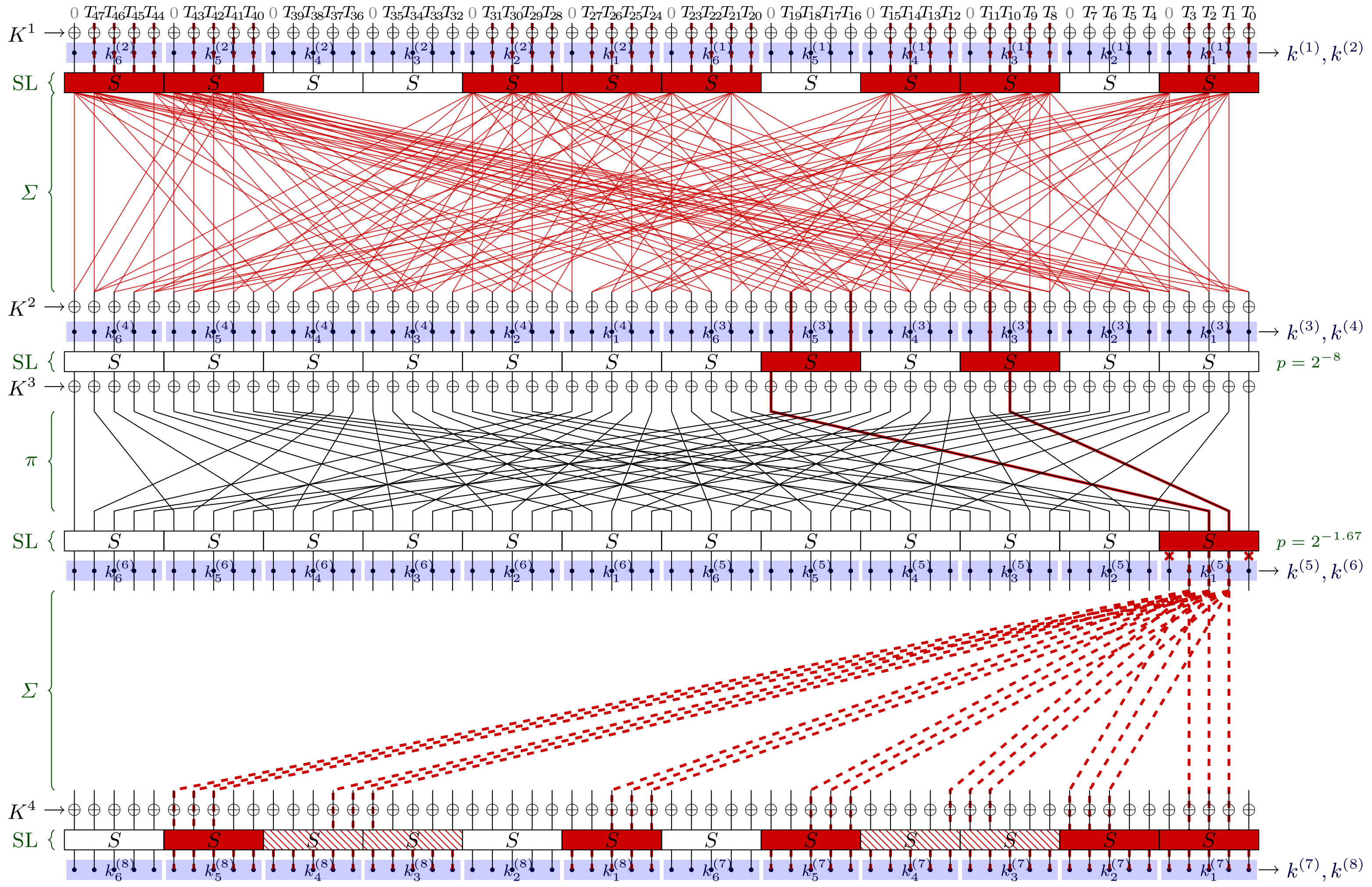
In the tweakey schedule, we consider pairs of tweaks so that there is a small number of active S-boxes in the second SL layer; therefore the last rounds will be similar to the 6-round distinguisher. This requires having many active S-boxes in the first SL layer, because the first Σ layer has a large branch number. We use the trail of Figure 6, with 8 active S-boxes in the first layer and 2 in the second layer. Moreover, this trail keeps both $k^{(4)}$ and $k^{(6)}$ inactive.

For the data path, we use the trail of Figure 4. We consider pairs of states which collide after the second round, and we partially decrypt them to obtain the corresponding plaintexts, which are queried to the oracle to check for a collision. As in the 6-round distinguisher, our trail assumes that the internal state collides at each round. This happens with relatively high probability when a tweak pair follows the trail in Figure 6: the subkeys $k_6^{(i)}$ are inactive between rounds 4 and 8, and the matrix is inactive in rounds 4 and 6. Experimentally, we measure the probability of collision as $1.7 \times 2^{-10} \approx 2^{-10} + 2^{-10.5}$, starting from pairs of tweaks following the first two S-box layers in the tweakey schedule and pairs of data colliding after the third round (see Section 5.2 for a more detailed analysis of the trail).

We first describe a basic attack with 2^{42} queries and high time complexity, and then we explain how to reduce the complexities below the security claims of the SCARF designers. As explained in section 3.2, we query the encryption-decryption oracle with a fixed tweak T^* for the decryption, and model it $\Pi(E_{K,T}(P))$ with Π a fixed but unknown permutation ($\Pi = E^{-1}_{K,T^*}$).

5.1 Basic Attack with 2^{42} Queries

The attack works as follows.



$K^1 \rightarrow$
$K^2 \rightarrow$
$K^3 \rightarrow$
$K^4 \rightarrow$
SL
Σ
π
$\rightarrow k^{(1)}, k^{(2)}$
$\rightarrow k^{(3)}, k^{(4)}$
$\rightarrow k^{(5)}, k^{(6)}$
$\rightarrow k^{(7)}, k^{(8)}$
$p = 2^{-8}$
$p = 2^{-1.67}$
S

Fig. 6: Related-tweak trail in the key-schedule of SCARF for the 8-round attack.

Table 3: Number of ordered pairs following the differential transition `0b0****` $\to \delta$, depending on the MSB of the key.

S-Box	Transition	Number of pairs $k_5 = 0$	Number of pairs $k_5 = 1$
1	`0b0****` $\to$ `0b10110`	6	6
3	`0b0****` $\to$ `0b11011`	8	8
4	`0b0****` $\to$ `0b01000`	8	8
6	`0b0****` $\to$ `0b10010`	6	6
7	`0b0****` $\to$ `0b10010`	6	6
8	`0b0****` $\to$ `0b10110`	6	6
11	`0b0****` $\to$ `0b11110`	6	6
12	`0b0****` $\to$ `0b11001`	8	8

1. We query and store the full codebooks for a set of tweaks where bits corresponding to the active S-boxes in the tweakey schedule trail ($T[0:3]$, $T[8:15], T[20:31], T[40:47]$) take all possible values, and the other bits are fixed to zero.
   This corresponds to 2^{42} queries, and we obtain values $\Pi(E_{K,T}(x))$.
2. We iterate over 2^{70} partial guesses of the key; we guess the full K^1 and the 10 bits of K^2 corresponding to the active S-boxes in the second tweakey schedule round, $K^2[10:14]$ and $K^2[20:24]$.
3. For each key guess, we filter out pairs of tweaks which follow the trail in the key schedule up to the second S-box layer.
4. For each such pair of tweaks (T, T'), we construct 2^{10} data pairs (P, P') that collide after the third round.
   Starting from a value (X_L, X_R) before the third round, we compute $\delta = G(X_L) \oplus G'(X_L) = \Delta M^{(2)} \times X_L$, where $\Delta M^{(2)}$ is the known key difference in the third round. We deduce the state value $(X'_L, X'_R) = (X_L, X_R \oplus \delta)$ such that (X_L, X_R) and (X'_L, X'_R) collide at the end of the third round. We partially decrypt (X_L, X_R) and (X'_L, X'_R) to obtain the plaintext pairs (P, P').
5. We count how many pairs collide, by comparing $\Pi(E_{K,T}(P))$ and $\Pi(E_{K,T'}(P'))$, which are part of the data queried in Step 1.

Success Probability. In order to analyze this attack, we first count how many pairs of tweaks (T, T') are expected for each key guess at Step 3.

For each active S-box in the first tweakey schedule layer, the input is the sum of a 5-bit key and a 4-bit tweak (padded with a zero bit). We consider a structure with all 2^4 tweaks, and we filter pairs with a fixed difference δ after the S-box. On average (for a random S-box and random difference), we expect $2^4 \times 2^4 \times 2^{-5} = 2^3$ ordered pairs of tweaks (T, T') with $S[k \oplus \text{expansion}(T)] \oplus S[k \oplus \text{expansion}(T')] = \delta$. However, taking into account the actual S-box and the trail, we have fewer pairs; Table 3 shows the number of valid pairs for each S-box, depending on the most significant bit of the key. In total we have $6^5 \times 8^3/2 = 1990656$ (un-ordered) pairs following the first S-box layer for each key.

In the second layer there are two active S-boxes, and the probability of the corresponding differential transition is 2^{-4} for both of them. Therefore we expect $1990656 \times 2^{-8} = 7776$ pairs (T, T') on average for each key. In Section 5.7, we show that in the worst case, the minimum number of pairs following the characteristic through the second S-box layer is $7740 \approx 2^{12.9}$.

In Step 4 we therefore build $2^{10} \times 2^{12.9} = 2^{22.9}$ samples $(T, P), (T', P')$ for each key guess. For a wrong key guess, we assume that the number of collisions follows a binomial distribution with $n = 2^{22.9}$ trials, and probability $q = 2^{-10}$. For the right key, the average probability of collision is around $p = 1.7 \times 2^{-10}$, but the distribution does not follow a binomial distribution. In experiments we observe a variance of $2^{15.3}$ instead of the expected $2^{13.7}$ for a binomial distribution. Instead of assuming a binomial distribution we implement the main step of the attack and measure experimentally the distribution of the number of collisions.

We distinguish the right key from the wrong keys according to the number of collisions detected. Using the threshold $t = (p + q)/2 = 1.35 \times 2^{-10}$, the probability of mistake is negligible. We can bound the probability of a wrong key having more collisions that the threshold using the Chernoff-Hoeffding theorem (with D the Kullback-Leibler divergence):

$$\Pr[B(n, q) \geq nt] \leq e^{-D(t\|q)n} \approx 2^{-628}$$

Therefore expect no wrong key to pass the threshold ($2^{70} \times 2^{-628} \lll 1$).

For the right key, we observe experimentally that the probability of being below the threshold is negligible (below 1/1000). We can also bound it using Chebyshev's inequality. We use the random variable X to denote the number of collisions. Experimentally we observe a standard deviation of $\sigma = 2^{7.6}$, therefore we have $np \approx nt + 14\sigma$ and we deduce:

$$\Pr[X \leq nt] \leq 1/14^2 \approx 0.5\%.$$

Complexity. If we implement this attack naively, the bottleneck of the attack is Step 3, were we filter tweak pairs following the trail for every key guess.

We optimize this step by observing that the pairs of tweaks following the first S-box layer depend only on K^1 and not K^2. Moreover, given a guess of K^1 and a tweak pair following the first S-box layer, we can easily determine the guesses of K^2 (restricted to the active S-boxes) compatible with the pair, because we know the output of the Σ layer, and the inputs to the S-box (the only pair following the differential trail). This is shown in Algorithm 1: the bottleneck is the loop over all K^1 guesses and queried tweaks of Line 4, evaluated $2^{32} \times 2^{60} = 2^{92}$ times.

5.2 Analysis of the Characteristic

We can analyse the 8-round characteristic in same way as we analysed the 6-round characteristic. In the key schedule, after filtering pairs of tweaks following the characteristic up to the second SL layer, we obtain

$$k^{(3)} \oplus k'^{(3)} = (\mathtt{0}, \mathtt{0}, \mathtt{a}, \mathtt{0}, \mathtt{9}, \mathtt{0}) \quad \text{and} \quad k^{(4)} = k'^{(4)}$$

With probability $5/16 = 2^{-1.67}$ the output difference of the active S-box in the third SL layer is only active on bits 1, 2 and 3. In this case we have:

$$k^{(5)} \oplus k'^{(5)} \in (\{\mathtt{2}, \mathtt{4}, \mathtt{8}, \mathtt{c}\}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}, \mathtt{0}) \quad \text{and} \quad k^{(6)} = k'^{(6)}$$

$$k^{(7)} \oplus k'^{(7)} = (*, *, *, *, *, \mathtt{0}) \quad \text{and} \quad k^{(8)} \oplus k'^{(8)} = (*, *, *, *, *, \mathtt{0})$$

In the data path we evaluate the distribution of the rank of the matrices $M \oplus M'$ based on the key schedule differences. We obtain an average probability $2/5 \approx 2^{-1.32}$ for round 5, and $2^{-4.023}$ for rounds 7 and 8. Therefore the probability that a pair of tweaks and plaintexts follows our trail is on average $2^{-1.67} \times 2^{-1.32} \times 2^{-4.023} \times 2^{-4.023} \approx 2^{-11}$. Finally we expect a probability of collision of roughly $2^{-10} + 2^{-11} = 1.5 \times 2^{-10}$ based on this trail. Since the experimental probability is 1.7×2^{-10}, this trail does not explain the full bias.

We point out that the 2^{10} pairs that we build for a given tweak pair do not behave independently. If the tweak pair follows the key schedule characteristic in the third S-box layer then all 2^{10} pairs have a high probability of colliding. Otherwise we expect them to collide with probability 2^{-10}. Moreover, the probability of the data characteristic depends on the rank of the matrices $M \oplus M'$ which are fixed by the tweak pair. This explains why the distribution of the number of collisions does not follow a binomial distribution. We expect that for each tweak pair, it follows a mixture of several binomial distributions, and the mixtures are summed to obtain the number of collisions.

Equivalent Keys. We observe that some keys cannot be distinguished by this attack. Indeed, if a pair of tweaks (T, T') follows the key schedule characteristic for a given key guess, then the input to the active S-boxes in the second SL layer must be exactly the pairs that satisfies the S-box transition. By flipping bits $K^2[11, 13]$ or $K^2[20, 23]$, we only swap the two elements of the pair, and the tweak pair is also valid for the new key guess. Therefore we obtain sets of 4 equivalent keys.

5.3 Reducing the Number of Queries

Since the attack has an overwhelming probability of success, we can reduce the query complexity and still keep a high success probability. More precisely, we select a smaller set of tweaks to query.

Following the analysis of Section 5.1, we count the number of tweak pairs compatible with the output difference of each individual S-box, considering a 4-bit tweak and a 5-bit key. In particular, we look for subspaces of 8 tweaks that still guarantee valid pairs for all keys; such subspaces exist for all active S-boxes in our trail.

Table 4: Number of ordered pairs following the differential transition $V \to \delta$, depending on the coset of inputs.

		Number of pairs			
S-Box	Transition	$\sigma_k = 0$ $k_5 = 0$	$\sigma_k = 0$ $k_5 = 1$	$\sigma_k = 1$ $k_5 = 0$	$\sigma_k = 1$ $k_5 = 1$
1	$V \to$ `0b10110`	2	2	2	2
6	$V \to$ `0b10010`	2	2	2	2
7	$V \to$ `0b10010`	2	2	2	2

We focus on S-boxes 1, 6 and 7, and we select the subspace of 8 tweaks with even Hamming weight:

$$V = \{\texttt{0b0000}, \texttt{0b0011}, \texttt{0b0101}, \texttt{0b0110}, \texttt{0b1001}, \texttt{0b1010}, \texttt{0b1100}, \texttt{0b1111}\}$$

For each S-box, we obtain 8 inputs $k \oplus \text{expansion}(T)$, corresponding to one of 4 cosets of V, depending on the most significant bit of k, and its Hamming weight σ_k. As shown in Table 4, we count the number of pairs in each coset of V leading to the desired output difference δ. For S-boxes 1, 6, and 7, this reduces the number of available pairs from 6 to 2. Overall, we reduce the number of queries from $2^{32} \times 2^{10} = 2^{42}$ to $2^{29} \times 2^{10} = 2^{39}$, and the average (respectively, minimal) number of tweaks compatible with a key decreases from $6^5 \times 8^3 \times 2^{-8}/2 = 7776$ (respectively 7740) to $2^3 \times 6^2 \times 8^3 \times 2^{-8}/2 = 288$ (respectively 284).

Success Probability. With this variant of the attack, we obtain $2^{10} \times 288 \approx 2^{18.2}$ samples $(T, P), (T', P')$ for each key guess. For a wrong key, we assume that the number of collisions follows a binomial distribution with $n = 2^{18.2}$ trials, and probability $q = 2^{-10}$. For the right key, we experimentally measure the distribution of the collision probability, and we set a threshold t so that the collision probability is higher than the threshold more than 99% of the time. We obtain $t = 1.4 \times 2^{-10}$ (see Figure 7).

Using this threshold t we have a significant advantage, with a small probability of missing the correct key. We estimate the advantage by evaluating the cumulative distribution function of the binomial distribution, giving tighter results than the Chernoff-Hoeffding theorem:

$$\Pr[B(n, q) > nt] \approx 2^{-34}$$

Therefore, running the attack with these parameters is expected to return a set of roughly $2^{70} \times 2^{-34} = 2^{36}$ candidates for the 70-bit key. With high probability (99%) this set contains the correct key.

5.4 Improving the Time Complexity

Algorithm 1 is inefficient due to two main reasons:

- For each key, the algorithm iterates over 2^{32} tweaks (respectively 2^{29} with less data) to identify a set of $2^{12.9}$ (respectively $2^{8.2}$) valid tweaks.
- The algorithm iterates over 2^{10} plaintexts to identify on average one collision.

We improve the complexity by precomputing some data, and reordering the steps of the attack, as shown in Algorithm 2.

First we observe that the valid pairs of tweaks for a given key guess can be computed efficiently. Indeed, we consider each S-box independently: we have an affine subspace of 5-bit inputs $k \oplus \text{expansion}(T)$, and we need to filter pairs (T, T') with a fixed output difference $S[k \oplus \text{expansion}(T)] \oplus S[k \oplus \text{expansion}(T')] = \delta$. As explained above, we obtain pairs of S-box inputs (u, u') with output difference δ depending on one or two key bits that define the affine subspace (see Tables 3 and 4). We deduce the corresponding tweak pairs as $T = u \oplus k, T' = u' \oplus k$, and just combine options from all S-boxes. For S-boxes 3, 4, 8, 11, and 12, we consider a set of tweaks of dimension 4 and only need the MSB of the 5-bit key; for S-Boxes 1, 6, and 7 we consider a set of tweaks of dimension 3 and need both the MSB of the key and its Hamming weight. Let us denote this set of 11 key bits as K^S.

After guessing K^S, we obtain $2^3 \times 6^2 \times 8^3/2 = 2^{16.2}$ pairs of S-box inputs (U, U') that satisfy the first S-box layer of the key schedule characteristic. In particular, the tweak difference $T \oplus T'$ is equal to the S-box input difference $U \oplus U$ and there are only $1^3 \times 3^2 \times 4^3 = 2^{9.2}$ different values $U \oplus U'$.

Secondly, instead of iterating over 2^{10} plaintext pairs for every valid tweak pair and checking for collisions, we precompute pairs of tweaks leading to collisions for every plaintext pair (P, P'). More precisely we create a table $\mathcal{L}$ indexed by the tweak difference $T \oplus T'$ and by the pair of plaintext (P, P'). After guessing K^S, there are only $2^{9.2}$ possible $T \oplus T'$ for pairs of tweaks following the key schedule characteristic. For a given $T \oplus T'$ and a given (P, P') we expect on average $2^{29} \times 2^{-10} = 2^{19}$ tweak pairs (T, T') leading to collisions. Therefore the total memory used is around $2^{9.2} \times 2^{19} \times 2^{19} = 2^{47.2}$ 32-bit words.

In order to exploit this precomputed data, we reorder the steps of attack: We first guess K^S and 20 bits of K^1 corresponding to inactive S-boxes, and we create counters indexed by the remaining key bits. Then we iterate over pairs of inputs to the first S-box layer (U, U'). This fixes the subkeys $k^{(1)}, k^{(2)}$ and we can generate pairs of plaintext (P, P') leading to collisions after the third round. For each such pair, we use the precomputed table to recover pairs of tweaks (T, T') with difference $U \oplus U'$ corresponding to collisions in the queried data. This fixes the full key $K^1 = U \oplus T$ and we can increment the corresponding counters (Algorithm 2, line 20).

We also need to compute the number of tweak pairs compatible with each guess. This number is not constant because the second S-box layer in the key schedule is probabilistic. We do this by iterating over all tweaks (line 14).

Complexity. The precomputation step of line 7 has complexity $2^{29} \times 2^{19} \times 2^{9.2} \times 2^{8+3} = 2^{68.2}$. The bottleneck of the attack is the loop on line 18, evaluated $2^{19} \times 2^{10} \times 2^{16.2} \times 2^{20} \times 2^{8+3} = 2^{76.2}$ times. This requires $2^{76.2}$ accesses to the precomputated table $\mathcal{L}$; we assume that a memory access has a complexity similar to the complexity of evaluating the block cipher[8].

5.5 Additional Filtering

The attack described so far returns on the average 2^{36} candidates for 70 key bits: K^1, $K^2[10:14]$, and $K^2[20:24]$. In order to filter these candidates we guess 20 additional key bits to increase the signal-to-noise ratio of the characteristic.

More precisely, we guess key bits $K^2[0:4]$, $K^2[35:39]$, $K^2[45:49]$ and $K^3[0,12,24,36,48]$. Combined with the previous guesses, we can now compute the input of the active S-box in the third SL layer of the tweakey schedule. Tweak pairs follow a valid transition with probability 5/16, and we filter valid tweak pairs for each key guess; we expect on average $288 \times 5/16 = 90$ tweak pairs. For tweaks pairs that follow the full tweakey schedule characteristic, the experimental probability to obtain a collision increases from $1.7 \times 2^{-10} \approx 2^{-10} + 2^{-10.5}$ to $3.2 \times 2^{-10} \approx 2^{-10} + 2^{-8.9}$.

We thus have three different distributions for the number of found collisions:

– Wrong values of the initial 70-bit guess have $q = 2^{-10}$ collision probability.
– Right guesses for the initial 70 bits combined with a wrong guess for the extra 20 bits have a collision probability of 1.7×2^{-10}.
– The right key guess has collision probability $p = 3.2 \times 2^{-10}$.

On average we have $n = 90 \times 2^{10} = 92160$ samples per key guess. This is not sufficient to recover a unique 90-bit key, but we use the additional filter to identify the correct 70-bit guess: if the guess is correct, there will be at least one guess of the additional 20 bits with a high bias p. On the other hand, if the 70-bit guess is wrong, any guess of the additional 20 bits should lead to a low bias q (we only distinguish the first and the third cases).

We assume the number of collisions for a wrong guess of the initial 70 bits follows a binomial distribution $B(n,q)$. For the right key, we use experiments to measure the probability distribution, and set a threshold so that more than 99% of the keys are above the threshold. As seen in Figure 8, this results in a threshold $t = 2.5 \times 2^{-10}$. With this threshold, we have:

$$\Pr[B(n,q) > nt] \approx 2^{-108}$$

Since this filtering is not independent of the previous part we cannot combine them, but we expect no candidate corresponding to a wrong guess of the initial 70 bits to pass the new filter ($2^{90} \times 2^{-108} \lll 1$). On the other hand, when combining both steps, we detect the correct key with probability at least 98%.

This step has a relatively small complexity compared to the initial filtering of the 70 key bits: We iterate over $2^{36} \times 2^{20}$ key candidates, and for each candidate we process $n = 2^{16.5}$ samples, for a total complexity of $2^{72.5}$.

[8] We point out that the table $\mathcal{L}$ is accessed sequentially by blocks of 2^{19} 32-bit words, therefore the latency should be reasonable.

Algorithm 1 Basic attack on 8 rounds, with 2^{42} queries

```
1: Query $\mathit{\Pi}(E_{K,T}(P))$                                        ▷ $2^{42}$
2: for all $K^1$ do                                                        ▷ 60 bits
3:     Initialize counters (pairs, colls) for $2^{10}$ $K^2$ candidates
4:     for all queried tweak $T$ do                                        ▷ $2^{32}$
5:         $Z \leftarrow \text{expansion}(T)$
6:         $Z' \leftarrow \text{SL}^{-1}\left(\text{SL}(Z \oplus K^1) \oplus \mathit{\Delta}^1\right) \oplus K^1$
7:         if $Z'$ is a valid expanded tweak then                          ▷ $2^{20.9}$
8:             $T' \leftarrow \text{expansion}^{-1}(Z')$
9:             $(T, T')$ satisfies the first S-Box layer
10:            Identify candidates $K^2$ compatible with $(T, T')$
11:            for all $(X_L, X_R)$ do                                     ▷ $2^{10}$
12:                $(X'_L, X'_R) \leftarrow (X_L, X_R \oplus \mathit{\Delta}M^{(2)} \times X_L)$
13:                Partially decrypt $(X, X')$ to obtain plaintexts $(P, P')$
14:                pairs[$K^2$] $\leftarrow$ pairs[$K^2$] $+ 1$
15:                if $\mathit{\Pi}(E_{K,T}(P)) = \mathit{\Pi}(E_{K,T'}(P'))$ then
16:                    colls[$K^2$] $\leftarrow$ colls[$K^2$] $+ 1$
17:    for all $K^2$ bits active in the second layer do                    ▷ 10 bits
18:        if colls[$K^2$]/pairs[$K^2$] $> t$ then
19:            $(K^1, K^2)$ is a key candidate
```

Algorithm 2 Improved attack on 8 rounds, with 2^{39} queries

```
1: Query $\mathit{\Pi}(E_{K,T}(P))$                                        ▷ $2^{39}$
2: for all $K^S$ (bits from active S-Boxes not covered by tweaks) do       ▷ 8+3 bits
3:     Build $2^{16.2}$ pairs of inputs $(U, U')$ for active S-Boxes of first SL layer
4:     Initialize array $\mathcal{L}$ of size $2^{9.2} \times 2^{19} \times 2^{19}$          ▷ $2^{47.2}$
5:     for all $U \oplus U'$ do                                            ▷ $2^{9.2}$
6:         for all plaintext pairs $(P, P')$ do                            ▷ $2^{19}$
7:             $\mathcal{L}[U \oplus U'][(P, P')] \leftarrow \{T \in \text{queries} : E_T(P) = E_{T \oplus U \oplus U'}(P')\}$   ▷ $2^{29}$
8:     for all $K^1$ corresponding to inactive S-Boxes do                  ▷ 20 bits
9:         Initialize counters (pairs, colls) for remaining 39 bits of key guess
10:        for all $(U, U')$ do                                            ▷ $2^{16.2}$
11:            70 bits of subkeys $k^{(1)}, k^{(2)}$ are known
12:            for all queried tweak $T$ do                                ▷ $2^{29}$
13:                Identify candidates $K^1, K^2$ compatible with $(T, T \oplus U \oplus U')$
14:                pairs[$K^1, K^2$] $\leftarrow$ pairs[$K^1, K^2$] $+ 2^{10}$
15:            for all $(X_L, X_R)$ do                                     ▷ $2^{10}$
16:                $(X'_L, X'_R) \leftarrow (X_L, X_R \oplus \mathit{\Delta}M^{(2)} \times X_L)$
17:                Partially decrypt $(X, X')$ to obtain plaintexts $(P, P')$
18:                for all $T \in \mathcal{L}[U \oplus U'][(P, P')]$ do        ▷ $2^{19}$
19:                    Identify candidates $K^1, K^2$ compatible with $(T, T \oplus U \oplus U')$
20:                    colls[$K^1, K^2$] $\leftarrow$ colls[$K^1, K^2$] $+ 1$
21:        for all remaining 39 bits of key do
22:            if colls[$K^1, K^2$]/pairs[$K^1, K^2$] $> t$ then
23:                $(K^1, K^2)$ is a key candidate
```

5.6 Experimental Verification

We have implemented the main steps of the attack with 2^{39} queries as a proof of concept: starting from the correct guess of K^1, we recover $K^2[10:14]$ and $K^2[20:24]$ in the first phase following Section 5.3, and we filter candidates in a second phase following Section 5.5. The code is available as supplementary material[9].

Out of 2048 experiments we observe:

- 19 cases where the key does not pass the threshold of the first step;
- 13 cases where the key passes the threshold of the first step, but not the second step;
- 2016 cases where the key passes both thresholds.

This matches the expected success rate of 98%.

Moreover, we observed no wrong keys passing the first threshold, apart from the 3 wrong keys equivalent to the correct key. Out of 2048 experiments with a wrong guess of K^1 we do not observe any key passing the first threshold either. This is coherent with the expected 34 bits of filtering.

5.7 Counting Minimal Compatible Tweak Pairs

To guarantee that a sufficient number of tweak pairs are available for every key guess in the attack, the minimal number of compatible tweak pairs over all keys was given in Sections 5.1 and 5.3. In this section we discuss the method that was used to derive these results. It is based on quasidifferential trails [4] and computes a tight lower bound on the number of compatible tweak pairs. We give an overview of quasidifferential trails in Appendix A.

As is shown in Theorem 1, the probability of a differential characteristic with inputs restricted to a subspace can be computed using quasidifferential trails. When the analyzed function is key dependent this gives rise to a function that maps keys from the key-space $\mathcal{K}$ to the respective probability of each key: $f : \mathcal{K} \to [0,1]$. When f is sufficiently simple, its minimum can be computed with off-the-shelf constraint optimization solvers, such as [11].

Theorem 1. *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function such that $F = F_r \circ \ldots \circ F_1$, let $V \subseteq \mathbb{F}_2^n$ be a subspace of the domain of F. The probability of a characteristic with differences $a_1, \ldots a_{r+1}$ and with input restricted to V is equal to:*

$$\Pr\left[\bigwedge_{i=1}^{r} F_i(\mathbf{x}_i + a_i) = F_i(\mathbf{x}_i) + a_{i+1}\right] = \sum_{u_1 \perp V} \sum_{u_2,\ldots,u_r} \prod_{i=1}^{r} D^{F_i}_{(u_{i+1},a_{i+1}),(u_i,a_i)}, \quad (1)$$

where $u_{r+1} = 0$, $\mathbf{x}_i = F_{i-1}(\mathbf{x}_{i-1})$ for $i \geq 2$ and $\mathbf{x}_1$ is sampled uniformly at random from V.

[9] https://github.com/SCARF-Cryptanalysis/SCARF-cryptanalysis

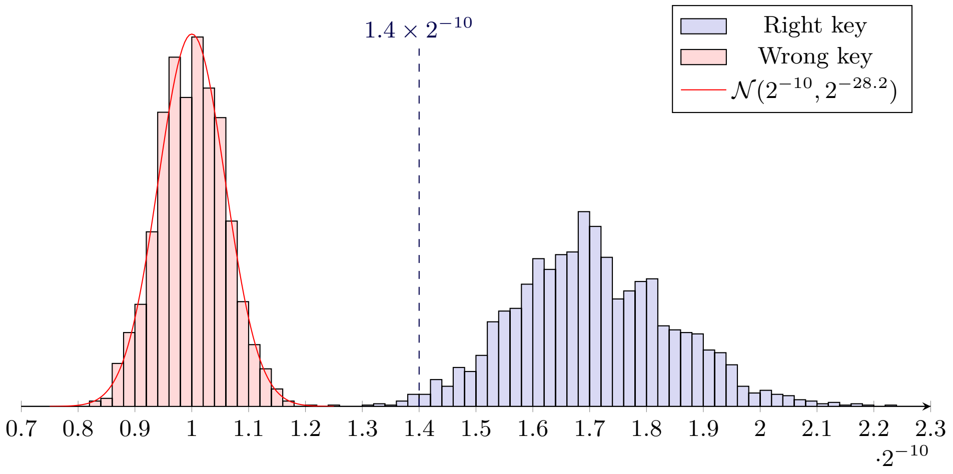


Fig. 7: Experimental verification of Section 5.3.
We simulate the attack 2^{12} times with random keys, and measure the observed collision probability for the right key and for a random (wrong) key. With a threshold of 1.4×2^{-10}, we miss less than 1% of right keys. For wrong keys, we obtain the distribution $B(2^{18.2}, 2^{-10})/2^{18.2}$ which can be approximated by $\mathcal{N}(2^{-10}, 2^{-28.2})$ (red curve).

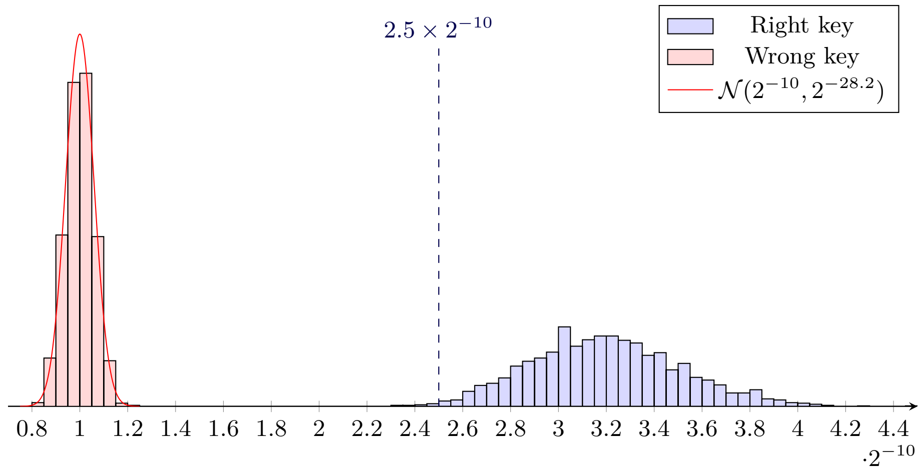


Fig. 8: Experimental verification of Section 5.5.
We simulate the attack 2^{12} times with random keys, and measure the observed collision probability for the right key and for a random (wrong) key. With a threshold of 2.5×2^{-10}, we miss less than 1% of right keys.

Proof. The result follows from the representation of the set of input pairs $S = \{(x, x + a_1) : x \in V\}$ in the quasidifferential basis. Let $f \in \mathbb{R}[\mathbb{F}_2^n \oplus \mathbb{F}_2^n]$ coincide with the probability density of the uniform distribution on S, i.e. $f[x, y] = |V|^{-1}\delta_V[x]\delta_a[y - x]$ and let $\tilde{f} = \mathscr{D}_n f$ be its representation in the dual quasidifferential basis. We have

$$\tilde{f}[u, a] = |V|^{-1} \sum_{\substack{(x,y)\in S \\ x+y=a}} (-1)^{u^\intercal x} = \delta_{a_1}(a)\delta_{V^\perp}(u),$$

which concludes the proof.

Note that multiplying Equation (1) by the size of V gives the number of ordered pairs which follow the characteristic. When $a_1 \in V$, the number of unique unordered pairs is half the number of ordered pairs.

Applying [4, Thm. 3.2] and Theorem 1 to the tweak-key schedule up to the addition with K_3 gives

$$\sum_{u_1 \in \bigoplus_{i=1}^{12} V_i^\perp} \sum_{u_3 \in \mathbb{F}_2^{60}} D^{SL}_{(u_2,a_2),(u_1,a_1)} D^{SL}_{(0,a_4),(u_3,a_3)} (-1)^{u_1^\intercal K^1 + u_3^\intercal K^2},$$

where (u_1, a_1), (u_2, a_2), (u_3, a_3) and $(0, u_4)$ are the mask-difference pairs of respectively the input and output of the first substitution layer and the input and output of the second substitution layer. V_i are the input spaces to each S-box, $a_3 = \Sigma(a_2)$, and $u_2 = \Sigma^\intercal(u_3)$.

To further model the truncated characteristic of Figure 6, it suffices to sum over all characteristics which follow the truncation pattern. However, the resulting number of quasidifferential trails with non-zero correlation is too large to enumerate. This can be remediated by reordering the sum to exploit that $D^{SL} = D^{S^{\otimes 12}}$ according to property 2 of [4, Thm. 3.2]:

$$f(K) = \sum_{u_3 \in \mathbb{F}_2^{60}} (-1)^{u_3^\intercal K^2} D^{SL}_{(0,a_4),(u_3,a_3)} \sum_{\substack{a \in \bigoplus_{i=1}^{12} V_i \\ u \in \bigoplus_{i=1}^{12} V_i^\perp}} (-1)^{u^\intercal K^1} D^{SL}_{(u_2,a_2),(u,a)}.$$

The second sum can now be rewritten as:

$$\prod_{i=1}^{R} \underbrace{\sum_{\substack{a_1 \in V_i \\ u_1 \in V_i^\perp}} (-1)^{u_1^\intercal[5i:5i+4] K^1[5i:5i+4]} D^S_{(u_2[5i:5i+4], a_2[5i:5i+4]),(u_1,a_1)}}_{\sum_{k\in\mathbb{F}_2^5} g(V_i, u_2[5i:5i+4], a_2[5i:5i+4], k)\delta_k(K^1[5i:5i+4])}.$$

By precomputing $g(V, v, b) = \sum_{u,a \in V^\perp \oplus V} (-1)^{u^\intercal k} D^S_{(v,b),(u,a)}$, $f(K)$ is simple enough to be minimized with a constraint minimization solver. Table 5 gives the results of these computations for the characteristic of Figure 6 and the subspace of Section 5.3 applied to an increasing number of S-boxes. The corresponding code is available as supplementary material[9].

Table 5: Expected and minimal number of unordered pairs that follow the characteristic of Figure 6 from the input space $\bigoplus_{i=1}^{12} V_i$ where $V_i = \{\texttt{0b0000}\}$ if the corresponding S-box is inactive, $V_i = \{\texttt{0b0000}, \texttt{0b0011}, \texttt{0b0101}, \texttt{0b0110}, \texttt{0b1001}, \texttt{0b1010}, \texttt{0b1100}, \texttt{0b1111}\}$ if $i \in S$ and $V_i = \mathbb{F}_2^4$ otherwise. The expected number of unordered pairs is the minimum number of unordered pairs through the first substitution layer multiplied by the expected probability of the second substitution layer.

S	$\{\}$	$\{1\}$	$\{1,3\}$	$\{1,3,7\}$	$\{1,3,7,8\}$
Expected	7776	2592	864	288	96
Minimal	7740	2580	860	284	92

6 Recovering the rest of the key

In the previous section we showed how to recover all of K^1 and 10 bits of K^2. We now show how to extract the rest of the key, assuming that all the bits of K^1, $K^2[10:14]$, and $K^2[20:24]$ are known. For this part of the attack we also use the differential trail given in Figure 6.

6.1 Finding 2^{16} Tweak Pairs Following the Given Trail

The first stage is to find 2^{16} tweak pairs (T_i, T_i') that follow the trail in Figure 6 with probability 1 up to the third S-box layer of the tweakey schedule. This can be readily done since we know the full K^1 and the key bits affecting the active S-boxes in the second round, but there is a small complication due to the fact that not all bits of K^1 are xored with tweak bits. For this reason we split the set of outputs from the S-box into two parts:

$$Z_0 = \{S[x] | x = \texttt{0b0****}\} \text{ and } Z_1 = \{S[x] | x = \texttt{0b1****}\}.$$

As $K^1[5i-1]$ is the most significant bit going into S-box i in the first S-box layer we know that the output value of S-box i must belong to $Z_{K^1[5i-1]}$, for $i = 1, \ldots, 12$.

We now fix pairs of 60-bit values $V_i = (v_1, \ldots, v_{12})$ and $V_i' = (v_1', \ldots, v_{12}')$, where each v_j and v_j' are 5-bit values. The V and V' represent two states after the first S-box layer in the tweakey schedule. As explained in Section 5.1, there are 1990656 valid pairs with the given output difference of the active S-boxes in the first round. For the passive S-boxes we select values $(v_2, v_5, v_9, v_{10}) = (v_2', v_5', v_9', v_{10}')$ from the set of 2^{16} different values from $Z_{K^1[9]} \times Z_{K^1[24]} \times Z_{K^1[44]} \times Z_{K^1[49]}$. In total we can therefore find $1990656 \cdot 2^{16} \approx 2^{36.9}$ pairs (V_i, V_i') that has the correct difference after the first S-box layer and will lead to the given input difference of the second S-box layer. The probability for following the characteristic through the two active S-boxes in this layer is 2^{-8}, so there will be $2^{28.9}$ (V_i, V_i')-pairs that follow the characteristic up to the input of the third S-box layer. Since all of K^1 and the key bits from K^2 affecting the active S-boxes in the second layer are known, we can filter out 2^{16} pairs (V_i, V_i') that with probability 1 will follow the characteristic in Figure 6 up to the third S-box layer. Going backwards through the first S-box layer we easily find the tweak pairs (T_i, T_i') that will end up in (V_i, V_i'), for $0 \leq i < 2^{16}$.

6.2 Guessing the Rest of K^2 and 10 Bits of K^3

At this point we ask for the full codebook for each of the tweaks T_i and T_i' for $0 \leq i < 2^{16}$. In total we are therefore making 2^{27} queries to the encryption oracle. The attack proceeds by guessing the remaining 50 bits of K^2. For each guess, we compute the values (W_i, W_i') after the second S-box layer, for each of the tweak pairs (T_i, T_i'). There are 10 bits of K^3 that affect the two S-boxes corresponding to $k_6^{(5)}$ and $k_6^{(6)}$. We now divide the set of 2^{16} (W_i, W_i') pairs and the corresponding (T_i, T_i') pairs into 2^{10} subsets $U_0, \ldots, U_{1023}$ according to these 10 bits. The pairs $((W_i, W_i'), (T_i, T_i'))$ are put in subset U_y if the value of W_i restricted to the 10 selected bits is y. Note that W_i' restricted to the same 10 bits is also y, since the characteristic guarantees the difference between W_i and W_i' is zero on these bits. We expect that each U_y will have 2^6 pairs each.

We continue by guessing the 10 bits of K_3 going into the S-boxes for $k_6^{(5)}$ and $k_6^{(6)}$. When the guessed 10-bit value is y, we only use the tweak pairs in U_y going forward. The (T_i, T_i') pairs in U_y will make sure that both $k_6^{(5)}$ and $k_6^{(6)}$ are zero for the current key guess.

6.3 Determining the Correct Guess by Counting Collisions

For each of the (expected) 2^6 tweak pairs (T_i, T_i') in the current subset U_y we can compute all round keys $k^{(1)}, k^{(2)}, k^{(3)}, k^{(4)}$. Set the state in the data path after the fourth round to $(0,0)$ (see Figure 1). For both T_i and T_i', decrypt this $(0,0)$-state to find the corresponding plaintexts P_i and P_i', both leading to the $(0,0)$-state after round 4. Since we know that $k_6^{(5)} = k_6^{(6)} = 0$ we know that the encryption of both P_i and P_i' will also have $(0,0)$-state after round 6, since $G(0) = 0$ regardless of the 25 key bits used in G, and because $S[0] = 0$.

There is only one active S-box in the third S-box layer in our trail in the tweakey schedule. With probability $5/16$, the pair (T_i, T_i') will follow the partial characteristic indicated for this S-box in Figure 6, i.e. there will be no difference in the least or the most significant bits of the S-box outputs. This implies that $k_6^{(7)}$ and $k_6^{(8)}$ will have the same values for both T_i and T_i' under the current key guess. When the partial characteristic after the third S-box layer is followed, we know that $E_{\tilde{K},T_i}(P_i) = E_{\tilde{K},T_i'}(P_i') = (k_6^{(8)}, S[k_6^{(7)}])$, where $\tilde{K}$ is the current key guess. That is, we have a pair of plaintexts and tweaks that collide.

For the correct key guess we therefore expect to have $5/16 \cdot 2^6 = 20$ colliding pairs from the (expected) 64 (P_i, P_i')-pairs. For a wrong key guess the probability of having a collision between (P_i, T_i) and (P_i', T_i') is 2^{-10}. The probability of having 11 or more collisions among the (expected) 64 pairs in U_y is

$$\sum_{i=11}^{64} \binom{64}{i} (2^{-10})^i (1-2^{-10})^{64-i} \approx 2^{-70.63}.$$

There are $2^{60} - 1$ wrong key guesses. For all wrong key guesses, the number of times we expect to see more than 11 collisions is $(2^{60} - 1)2^{-70.63} = 2^{-10.63}$. So with probability less than 1/1000 do we expect to see more than 10 collisions for wrong key guesses. The large gap in the number of collisions for right and wrong key guesses means we can clearly distinguish the correct key guess from all the wrong ones, even if the number of pairs in the U_y's deviates slightly from the expected 64. For instance, with 50 pairs in a U_y, we expect 15.63 collisions for the correct key guess, while the probability of seeing more than 9 collisions among the wrong key guesses is $2^{-6.79}$. Hence the key guess returning the most collisions will be the correct K^2 and the correct 10 selected bits from K^3.

Complexity The complexity of the part of the attack given in this section is dominated by guessing the remaining 50 bits of K^2, and visiting all 2^{16} pairs of tweaks for each guess. This gives a time complexity of 2^{66}.

6.4 Recovering the rest of K^3

After learning K^2 and 10 bits of K^3 we can continue the attack by guessing the remaining part of K^3. For each guess, we first find a set of 8 tweak pairs (T_i, T_i') for $1 \leq i \leq 8$ that give equal values of $k_6^{(7)}$ and $k_6^{(8)}$. This can be done by first fixing T_i to an arbitrary value and computing Y_i, the 60-bit value output from the last Σ in the tweakey schedule. Next, we try random values for T_i' and compute the corresponding Y_i', until $Y_i'[25:29] = Y_i[25:29]$ and $Y_i'[55:59] = Y_i[55:59]$. These 10 bits of Y_i and Y_i' determine $k_6^{(7)}$ and $k_6^{(8)}$ and will therefore give equal values for these sub-keys. We expect to try 2^9 different T_i' before a match is found, so repeating this search 8 times gives a total complexity for finding the (T_i, T_i')-pairs of $2^{50} \cdot 2^9 \cdot 8 = 2^{62}$.

For each guess of K^3 we can compute all round keys $k^{(1)}, \ldots, k^{(6)}$. For each of the tweak pairs (T_i, T_i') we can then set the $(0,0)$-state after round 6 and decrypt to find the corresponding plaintexts (P_i, P_i') that will give the $(0,0)$-state after six rounds of encryption. Since we know $k_6^{(7)}$ and $k_6^{(8)}$ will be the same for both T_i and T_i', we know that $E_{K,T_i}(P_i) = E_{K,T_i'}(P_i')$ for all $1 \leq i \leq 8$ when the remaining 50 bits of K^3 are guessed correctly. That is, we will see 8 collisions in the ciphertexts for the 8 (P_i, P_i')-pairs we have constructed. For wrong key guesses, we get a collision in each pair with probability 2^{-10}. Therefore, the probability of having 8 collisions among the wrong key guesses is $2^{-80} \cdot 2^{50} = 2^{-30}$. Take the guess giving 8 collisions as the correct value for K^3.

6.5 Recovering K^4

When K^1, K^2, and K^3 have been recovered, the last 60 bits in K^4 can be recovered as follows. Guess $K^4[0:29]$, the half of K^4 that determines $k^{(7)}$. For each guess, find 8 tweak pairs (T_i, T_i') giving equal $k_6^{(8)}$. For each pair, we expect to try 2^4 tweak pairs before a valid pair is found. The total complexity for this part is therefore $2^{30} \cdot 2^4 \cdot 2^3 = 2^{37}$.

For each key guess and each of the 8 (T_i, T_i') pairs, set the cipher state after round 7 to $(0, a)$ for some arbitrary value a. Decrypt this state with the current key guess and T_i, T_i' to find the corresponding plaintexts (P_i, P_i'). We know that $E_{K,T_i}(P_i) = E_{K,T_i'}(P_i')$ for all $1 \le i \le 8$ when we have guessed the 30 bits of $K^4[0:29]$ correctly. All key guesses will have a collision in the right half of the ciphertext, but wrong key guesses will only collide in the left half with probability 2^{-5}. The probability of having 8 collisions among the wrong key guesses is therefore $2^{-40} \cdot 2^{30} = 2^{-10}$. So we expect only one guess for $K^4[0:29]$ to collide in all eight pairs, and take that as the correct value.

Finally, the 30 remaining bits of K^4 can be guessed for a complete key candidate K with complexity 2^{30}, which is verified against the known code book for some tweak.

7 Conclusions

We have demonstrated that using tweak pairs exhibiting a well-chosen difference can be used to introduce a significant bias in the probability that two ciphertexts collide. We illustrated this approach in a distinguishing attack on 6-round SCARF with query and time complexities of 2^{30}. This distinguisher effectively works by counting the number of collisions given by the collision oracle and thus serves as an example of an effective distinguisher in that model.

We next demonstrate how this approach can be extended to a key-recovery attack on full-round SCARF in the encryption-decryption model. With a query complexity of 2^{39} and a time complexity of 2^{77}, this attack shows that full-round SCARF does not achieve the designers' intended security level in the encryption-decryption model. Whether SCARF is fully secure in the more conservative collision model remains an open problem.

Acknowledgements. The authors would like to thank the Lorentz Center workshop *Beating Real-Time Crypto: Solutions and Analysis* and its scientific organizers Shahram Rasoolzadeh, Christina Boura, Maria Eichlseder, Bart Mennink and Yosuke Todo for making this collaboration possible. This work was partially supported by the Research Council KU Leuven, C16/18/004 through the C1 on New Block Cipher Structures. In addition, this work was supported by CyberSecurity Research Flanders with reference number VR20192203. This work was also partially supported by the project *Cryptanalyse* from PEPR Cybersécurité (22-PECY-0010).

References

1. Beyne, T.: A geometric approach to linear cryptanalysis. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 36–66. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_2
2. Beyne, T.: A geometric approach to symmetric-key cryptanalysis. Ph.D. thesis, KU Leuven (June 2023)
3. Beyne, T.: Linear and differential cryptanalysis (2023), https://cryptanalysis.info/static/personal/sac2023/slides_school.pdf, Tutorial 3 at the SAC summer school
4. Beyne, T., Rijmen, V.: Differential cryptanalysis in the fixed-key model. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part III. LNCS, vol. 13509, pp. 687–716. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15982-4_23
5. Beyne, T., Verbauwhede, M.: Ultrametric integral cryptanalysis. Cryptology ePrint Archive, Report 2024/722 (2024), https://eprint.iacr.org/2024/722
6. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO'90. LNCS, vol. 537, pp. 2–21. Springer, Berlin, Heidelberg (Aug 1991). https://doi.org/10.1007/3-540-38424-3_1
7. Boura, C., Rasoolzadeh, S., Saha, D., Todo, Y.: Multiple-tweak differential attack against SCARF. Cryptology ePrint Archive, Report 2024/1408 (2024), https://eprint.iacr.org/2024/1408, accepted to ASIACRYPT 2024
8. Canale, F., Güneysu, T., Leander, G., Thoma, J.P., Todo, Y., Ueno, R.: SCARF - A low-latency block cipher for secure cache-randomization. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 1937–1954. USENIX Association (Aug 2023)
9. Chen, S., Hu, K., Liu, G., Niu, Z., Tan, Q.Q., Wang, S.: Meet-in-the-middle attack on 4+4 rounds of SCARF under single-tweak setting. Cryptology ePrint Archive, Report 2024/1270 (2024), https://eprint.iacr.org/2024/1270
10. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Berlin, Heidelberg (Apr 2002). https://doi.org/10.1007/3-540-45473-X_13
11. Perron, L., Didier, F.: CP-SAT, https://developers.google.com/optimization/cp/cp_solver/
12. Qureshi, M.K.: CEASER: mitigating conflict-based cache attacks via encrypted-address and remapping. In: 51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2018, Fukuoka, Japan, October 20-24, 2018. pp. 775–787. IEEE Computer Society (2018). https://doi.org/10.1109/MICRO.2018.00068, https://doi.org/10.1109/MICRO.2018.00068
13. Saileshwar, G., Qureshi, M.K.: MIRAGE: mitigating conflict-based cache attacks with a practical fully-associative design. In: Bailey, M.D., Greenstadt, R. (eds.) 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021. pp. 1379–1396. USENIX Association (2021), https://www.usenix.org/conference/usenixsecurity21/presentation/saileshwar
14. Wang, Z., Lee, R.B.: New cache designs for thwarting software cache-based side channel attacks. In: Tullsen, D.M., Calder, B. (eds.) 34th International Symposium on Computer Architecture (ISCA 2007), June 9-13, 2007, San Diego, California, USA. pp. 494–505. ACM (2007). https://doi.org/10.1145/1250662.1250723, https://doi.org/10.1145/1250662.1250723

15. Werner, M., Unterluggauer, T., Giner, L., Schwarz, M., Gruss, D., Mangard, S.: Scattercache: Thwarting cache attacks via cache set randomization. In: Heninger, N., Traynor, P. (eds.) 28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019. pp. 675–692. USENIX Association (2019), https://www.usenix.org/conference/usenixsecurity19/presentation/werner

A Differential Cryptanalysis

The presentation below follows the geometric approach to differential cryptanalysis by Beyne and Rijmen [4], although we use the updated description of the geometric approach from [5]. For simplicity, we will restrict this description and the subsequent derivation of differential cryptanalysis to vectorial Boolean function.

The geometric approach studies vectorial Boolean functions, $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, by extending them to linear functions from the free k-vector space on $\mathbb{F}_2^n$ to the free k-vector space on $\mathbb{F}_2^m$. This is the push forward operator of F. In the context of differential cryptanalysis of vectorial Boolean functions, it suffices to take k as $\mathbb{R}$. Additionally, we index vectors in this free vector space with square brackets. The notation δ_x is used for the standard basis vectors, and they are defined as $\delta_x[y] = 1$ if and only if $x = y$ and 0 otherwise.

Definition 1 (Pushforward Operator and Transition Matrix). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. The pushforward operator of F is the unique linear operator $T^F : \mathbb{R}[\mathbb{F}_2^n] \to \mathbb{R}[\mathbb{F}_2^m]$ defined by $\delta_x \mapsto \delta_{F(x)}$ for all $x \in \mathbb{F}_2^n$. The transition matrix of F is the matrix representation of T^F in the standard basis of $\mathbb{R}[\mathbb{F}_2^n]$ and $\mathbb{R}[\mathbb{F}_2^m]$.*

The pushforward operator can be interpreted as propagating information, in the form of elements of the free vector space, forwards through a function. Similarly, we can define the pullback operator of a function which propagates information backwards. It is the dual of the pushforward operator and maps the k-valued functions on $\mathbb{F}_2^m$ to the k-valued functions on $\mathbb{F}_2^n$. We use round brackets for function evaluation. The notation δ^x is used for the standard basis vectors, and they are defined as $\delta^x(y) = 1$ if and only if $x = y$ and 0 otherwise. Note that we identify the dual of the free k-vector space on a set with the k-valued functions on the same set by extending the function evaluation linearly to the k-vector space. That is $h(g) = \sum_{x \in \mathbb{F}_2^n} h(x)g[x]$ with $g \in \mathbb{R}[\mathbb{F}_2^n]$ and $h : \mathbb{F}_2^n \to \mathbb{R}$. Alternatively we write $h \in \mathbb{R}^{\mathbb{F}_2^m}$.

Definition 2 (Pullback Operator [2, Def. 2.8]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. The pullback operator of F is the unique linear operator $T^{F^\vee} : \mathbb{R}^{\mathbb{F}_2^m} \to \mathbb{R}^{\mathbb{F}_2^n}$ defined by $T^{F^\vee}(\delta^x) = \delta^x \circ F$ for all $x \in \mathbb{F}_2^m$.*

The pushforward and pullback operator of the composition or concatenation of functions can also be expressed as the product and tensor product of the individual functions, respectively.

Theorem 2 (Properties of the Pushforward and Pullback Operator [2, Thm. 2.4]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. The pushforward operator T^F and pullback operator $T^{F^\vee}$ satisfy the following properties:*

1. *if $F(x_l \| \dots \| x_1) = F_l(x_l) \| \dots \| F_1(x_1)$ then $T^F = \bigotimes_{i=1}^l T^{F_i}$ and $T^{F^\vee} = \bigotimes_{i=1}^l T^{F_i{}^\vee}$,*
2. *if $F = F_r \circ \dots \circ F_1$ then $T^F = T^{F_r} \cdots T^{F_1}$ and $T^{F^\vee} = T^{F_1{}^\vee} \cdots T^{F_r{}^\vee}$.*

Many cryptographic properties can be expressed by combining forward and backward propagation. For example, in differential cryptanalysis [6], we are interested in the probability that an input pair $(x, x+a)$ results in an output pair $(x', x'+b)$ for some difference $a \in \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^m$ and a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$. Since we sample x uniformly at random from the domain of F, we can encode this as a vector $g \in \mathbb{R}[\mathbb{F}_2^n \oplus \mathbb{F}_2^n]$ where $g[x, y] = 2^{-n}\delta_a[y - x]$. Similarly, the check for the output difference can be encoded as $h \in \mathbb{R}^{\mathbb{F}_2^m \oplus \mathbb{F}_2^m}$ where $h(x, y) = \delta^b(y - x)$. The probability of the differential is then equal to $h\left((T^F \otimes T^F)g\right)$. Note that g and h are defined on the sets $\mathbb{F}_2^n \oplus \mathbb{F}_2^n$ and $\mathbb{F}_2^m \oplus \mathbb{F}_2^m$ respectively, because differential cryptanalysis requires pairs of values.

One of the strengths of the geometric approach is that we can analyse the transition matrix in any basis. Such a basis can be chosen to simplify the description of the properties of interest and/or a part of the cipher, such as the key addition. A choice of basis that describes differential properties are the indicators for the difference, $\delta^b(y - x)$. However, this basis does not span the full space and needs to be extended. The quasidifferential basis is the extension of these difference indicators with the additive character basis used in linear cryptanalysis [1]. This choice additionally diagonalizes the transition matrix of the constant addition, which simplifies the analysis of key additions.

Definition 3 (Quasidifferential Basis [4, Def. 3.1]). *For any $u, a \in \mathbb{F}_2^n$ the function $\beta^{u,a} : \mathbb{F}_2^n \oplus \mathbb{F}_2^n \to \mathbb{R}$ is defined as*

$$\beta^{u,a}(x, y) = \chi^u(x)\delta^a(y - x),$$

where $\chi^u(x) = (-1)^{u^\intercal x}$. The set of all $\beta^{u,a}$ defines the quasidifferential basis.

Because the quasidifferential basis is orthogonal [4, Thm. 3.1], its dual basis consists of the vectors $\beta_{u,a}[x, y] = 2^{-n}\beta^{u,a}(x, y)$. Let $\mathscr{D}_n$ be the change of basis transformation with respect to this dual basis. The quasidifferential transition matrix can then be defined as the change of basis transformation of $T^F \otimes T^F$ to this dual basis.

Definition 4 (Quasidifferential Transition Matrix [4, Def. 3.1]). *For a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, let $D^F = \mathscr{D}_m(T^F \otimes T^F)\mathscr{D}_n^{-1}$. The quasidifferential transition matrix is the coordinate representation of D^F with respect to the standard bases of $\mathbb{R}[\mathbb{F}_2^n \oplus \mathbb{F}_2^n]$ and $\mathbb{R}[\mathbb{F}_2^m \oplus \mathbb{F}_2^m]$.*

The properties of the transition matrix from Theorem 2 carry over to the quasidifferential transition matrix. Additionally, the quasidifferential transition matrices of affine functions have a simple description.

Theorem 3 (Properties of Quasidifferential Transition Matrices [4, Thm. 3.2]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function. The quasidifferential transition matrix of F, D^F has the following properties:*

1. *if $F(x_l \| \dots \| x_1) = F_l(x_l) \| \dots \| F_1(x_1)$, then $D^F = \bigotimes_{i=1}^{l} D^{F_i}$,*
2. *if $F = F_r \circ \dots \circ F_1$ then $D^F = D^{F_r} \dots D^{F_1}$,*
3. *if $F(x) = x + t$, then $D^F_{(v,b),(u,a)} = (-1)^{u^\mathsf{T} t} \delta_u(v) \delta_a(b)$,*
4. *if F is a linear function, $D^F_{(v,b),(u,a)} = \delta_u(F^\mathsf{T}(v)) \delta_b(F(a))$.*

At the SAC 2023 summer school, Beyne gave an alternative description of the quasidifferential transition matrix which highlights the information that a single entry propagates [3]. The entries of the quasidifferential transition matrix are equal to the probability of the difference a to b through F times the correlation of the linear approximation $u^\mathsf{T} x = v^\mathsf{T} F(x)$ conditioned on the right pairs of the difference:

$$\begin{aligned} D^F_{(v,b),(u,a)} &= \left(2 \Pr\left[u^\mathsf{T} \mathbf{x} = v^\mathsf{T} F(\mathbf{x}) \mid F(\mathbf{x} + a) = F(\mathbf{x}) + b\right] - 1\right) \\ &\quad \times \Pr\left[F(\mathbf{x} + a) = F(\mathbf{x}) + b\right], \end{aligned} \tag{2}$$

where $\mathbf{x}$ is a random vector uniformly distributed on $\mathbb{F}_2^n$.

When $F = F_r \circ \dots \circ F_1$, property 2 of Theorem 3 gives rise to a trail-based approach to computing the probability of a differential:

$$\Pr\left[F(\mathbf{x} + a) = F(\mathbf{x}) + b\right] = D^F_{(0,b),(0,a)} = \sum_{a_2, \dots a_r, u_2, \dots u_r} \prod_{i=1}^{r} D^{F_i}_{(u_{i+1}, a_{i+1}),(u_i, a_i)},$$

where $u_1 = u_{r+1} = 0$, $a_1 = a$ and $a_{r+1} = b$. From the alternative description of Equation (2) it can be seen that, for a fixed sequence of differences, the quasidifferential trail with the all-zero masks coincides with the corresponding differential characteristic. The correlation of this quasidifferential trail is equal to the expected differential probability of the characteristic, as it would be derived through ordinary differential cryptanalysis:

$$\Pr\left[\bigwedge_{i=1}^{r} F_i(\mathbf{x}_i + a_i) = F(\mathbf{x}_i) + a_{i+1}\right] = \prod_{i=1}^{r} D^F_{(0, a_{i+1}),(0, a_i)},$$

where all $\mathbf{x}_i$ are independently uniform on the domain of their corresponding F_i.

All other quasidifferential trails following the same characteristic can be seen as corrections to the expected differential probability based on probabilistic linear relations on the pairs that satisfy the characteristic. By summing over all quasidifferential trails for a fixed characteristic, the exact probability of that characteristic can be computed.

Theorem 4 (Exact Probability of a Differential Characteristic [4, Thm. 4.1]). *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a vectorial Boolean function such that $F = F_r \circ \dots \circ F_1$.*

The probability of a differential characteristic $a_1, \ldots, a_{r+1}$ *is equal to the sum of the correlations of all quasidifferential trails with the same differences:*

$$\Pr\left[\bigwedge_{i=1}^{r} F_i(\mathbf{x}_i + a_i) = F(\mathbf{x}_i) + a_{i+1}\right] = \sum_{u_2,\ldots,u_r} \prod_{i=1}^{r} D^F_{(u_{i+1},a_{i+1}),(u_i,a_i)},$$

where $u_1 = u_{r+1} = 0$, $\mathbf{x}_i = F_{i-1}(\mathbf{x}_{i-1})$ *for* $i > 1$ *and* $\mathbf{x}_1$ *uniform random on the domain of* F.