Note of Multidimensional MITM Attack on 25-Round TWINE-128

Long Wen[1], Meiqin Wang[1*], Andrey Bogdanov[2*], Huaifeng Chen[1]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
`longwen@mail.sdu.edu.cn, mqwang@sdu.edu.cn`
[2] Technical University of Denmark, Denmark
`anbog@dtu.dk`

Abstract. TWINE is a lightweight block cipher proposed in SAC 2012 by Suzaki *et al.*. TWINE operates on 64-bit block and supports 80 or 128-bit key, denoted as TWINE-80 and TWINE-128 respectively. TWINE has attracted some attention since its publication and its security has been analyzed against several cryptanalytic techniques in both single-key and related-key settings. In the single-key setting, the best attack so far is reported by Boztaş *et al.* at LightSec'13, where a splice-and-cut attack on 21-round TWINE-128 and a multidimensional meet-in-the-middle (MITM) attack on 25-round TWINE-128 are presented. Yet, the evaluation of the time complexity of the multidimensional MITM attack on 25-round TWINE-128 is somehow controversial in the way we understand. We here describe the attack in detail and explains our concerns about the time complexity of the attack. And it turns out that the multidimensional MITM attack on 25-round TWINE-128 may have a time complexity higher than exhaustive search.

Keywords: Block Ciphers, Cryptanalysis, TWINE, Multidimensional MITM Attack

1 Description of TWINE

TWINE [1] is a 64-bit block cipher supporting two key lengths, 80 or 128 bits (denoted as TWINE-80 and TWINE-128 respectively). The two versions of TWINE algorithm differ only in the key schedule. The global structure of TWINE is a variant of Type 2 generalized Feistel structure with 16 4-bit nibbles. TWINE's round function consists of a nonlinear layer using 4×4 S-boxes and a diffusion layer permuting the 16 nibbles. The r-th $(1 \leq r \leq 36)$ round of TWINE is illustrated in Figure 1, where we use $X_r^i, 0 \leq i \leq 15$ to denote the 16 input nibbles into round r. Each round function takes a 32-bit subkey $RK_r, 1 \leq r \leq 36$. The i-th nibble of RK_r is denoted as $RK_r^i, 0 \leq i \leq 7$.

The key schedules produce 36 32-bit subkeys $RK_r, 1 \leq r \leq 36$ from the key κ. κ could be 80 or 128 bits depending on which version of TWINE we are dealing with. For details of the key schedule, we refer to [1].

* Corresponding authors.

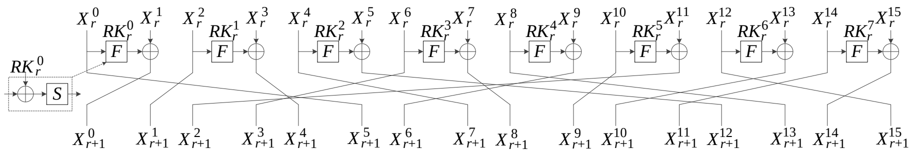
Fig. 1: r-th round of TWINE

2 Multidimensional MITM Attack on 25-Round TWINE-128 [4]

In [4], Boztaş *et al.* combined Splice-and-Cut technique [2] and multidimensional MITM technique [3] and mounted a key recovery attack on 25-round TWINE-128. The attack requires 2^{48} chosen plaintexts, 2^{122} encryptions and 2^{125} blocks memory. The idea of their attack is interesting. However, after a careful evaluation on the attack procedure, we find some flaws in term of the time complexity of the attack on 25-round TWINE-128 [4]. The actual time complexity of their attack on 25-round TWINE-128 should be higher than what is claimed and unfortunately even higher than exhaustive search.

In the following, we firstly introduce the two dimensional MITM (2D-MITM) technique used in [4]. Then we dive into the details of the attack on 25-round TWINE-128 [4]. Finally, we revaluate the time complexity of the attack.

2.1 2D-MITM Technique

The main idea of the 2D-MITM attack is that the attacker splits the target cipher into two sub-ciphers and performs MITM attack on each of the sub-cipher by guessing an intermediate stage g, see Figure 2. Denote the common key bits of subkey k_1 and k_2 as k_{c_1} and the common key bits of subkey k_3 and k_4 as k_{c_2}, the two dimensional MITM attack can be proceeded as follows.

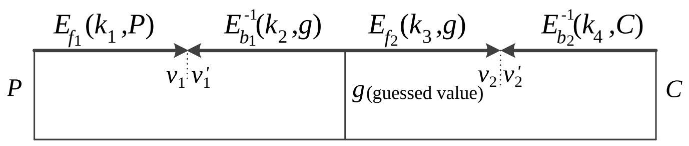

Fig. 2: overview of 2D-MITM attack

1. Compute $v_1 = E_{f_1}(k_1, P)$ for all values of k_1, and put all values of k_1 into a table T_1 indexed by $v_1||k_{c_1}$, each entry of which is a set of certain k_1.
2. Compute $v'_2 = E^{-1}_{b_2}(k_4, C)$ for all values of k_4, and put all values of k_4 into a table T_2 (similar as T_1) indexed by $v'_2||k_{c_2}$.
3. For each possible guess of g:

(a) Compute $v'_1 = E^{-1}_{b_1}(k_2, g)$ for all values of k_2. Use v'_1 and k_{c_1} to find matched entries of T_1. Put all matched (k_1, k_2) into a table Q.
(b) Compute $v_2 = E_{f_2}(k_3, g)$ for all values of possible k_3. Use v_2 and k_{c_2} to find matched entries of T_2. Check whether (k_3, k_4) are also matched entries of Q. If so, do brute-force testing on the matched key tuple (k_1, k_2, k_3, k_4) with other PC pairs. If the key tuple survives all test-s, output the key tuple as the correct key and stop the attack.

2.2 Overview of the Attack on 25-Round TWINE-128 [4]

The 2D-MITM attack combined with Splice-and-Cut technique is applied on 25-round TWINE-128, and the attack framework is shown in Figure 3.

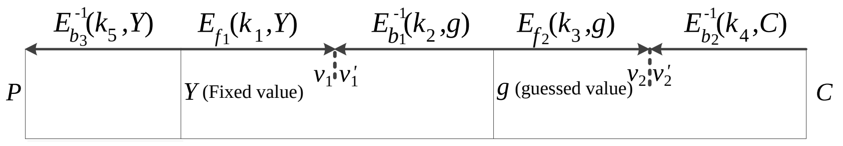

Fig. 3: overview of Boztaş *et al.*'s attack on 25-round TWINE-128

TWINE-128's key schedule is taken into consideration to make the attack work. RK_1 is extracted directly from κ, then κ is updated to produce the next subkey. If RK_i is extracted from the i-th κ state (denoted as κ_i), then it is the relation between subkey nibbles and κ_8 that are taken into consideration in the attack on 25-round TWINE-128.

The details of the attack on 25-round TWINE-128 are illustrated in Figure 3 and Figure 4. In both figures, we show different partial encryption (or decryption) phase in different colors and the states nibbles and F-functions that are involved in the partial encryption (or decryption) are shown in bold. Since κ_8^1 and κ_8^{22} are the two special key nibbles considered, if a F-function's input subkey is related to κ_8^1 or κ_8^{22}, we put the number '1' or '22' beneath the F-function in both figures.

To be more concrete, $Y = X_1^{2,6,14} || X_2^{0,2,6,7,8,10,12} || X_3^{6,12,14} || X_4^2 || X_5^{6,13}$, $g = X_{15}^{0,\dots,7} || X_{16}^{2,6,9,10,11,13,14,15}$. The first check point is $v_1 = v'_1 = X_{13}^{4,5,6,9,10,11,13,14,15}$. The second check point is $v_2 = v'_2 = X_{18}^{2,4,5,6,7,10,11,14}$. If we denote the bit length of a variable with $|\cdot|$, then $|v_1| = |v'_1| = 36$ and $|v_2| = |v'_2| = 32$. k_1 used in E_{f_1} has nothing to do with κ_8^1 and κ_8^{22} is not involved in $E^{-1}_{b_2}$ and $E^{-1}_{b_3}$. Thus $|k_1| = 124$ and $|k_4| = 124$. In $E^{-1}_{b_1}$ and E_{f_2}, there are respective 15 and 16 F-functions involved, this makes $|k_2| = 60$ and $|k_3| = 64$. According to TWINE-128's key schedule, there are 48 common bits between k_1 and k_2 and 48 common bits between k_3 and k_4, meaning that $|k_{c_1}| = |k_{c_2}| = 48$. The attack on 25-round TWINE-128 is proceeded as follows.

1. For all k_1, compute $v_1 = E_{f_1}(k_1, Y)$ and construct T_1 containing all values of k_1 indexed by $v_1 || k_{c_1}$.

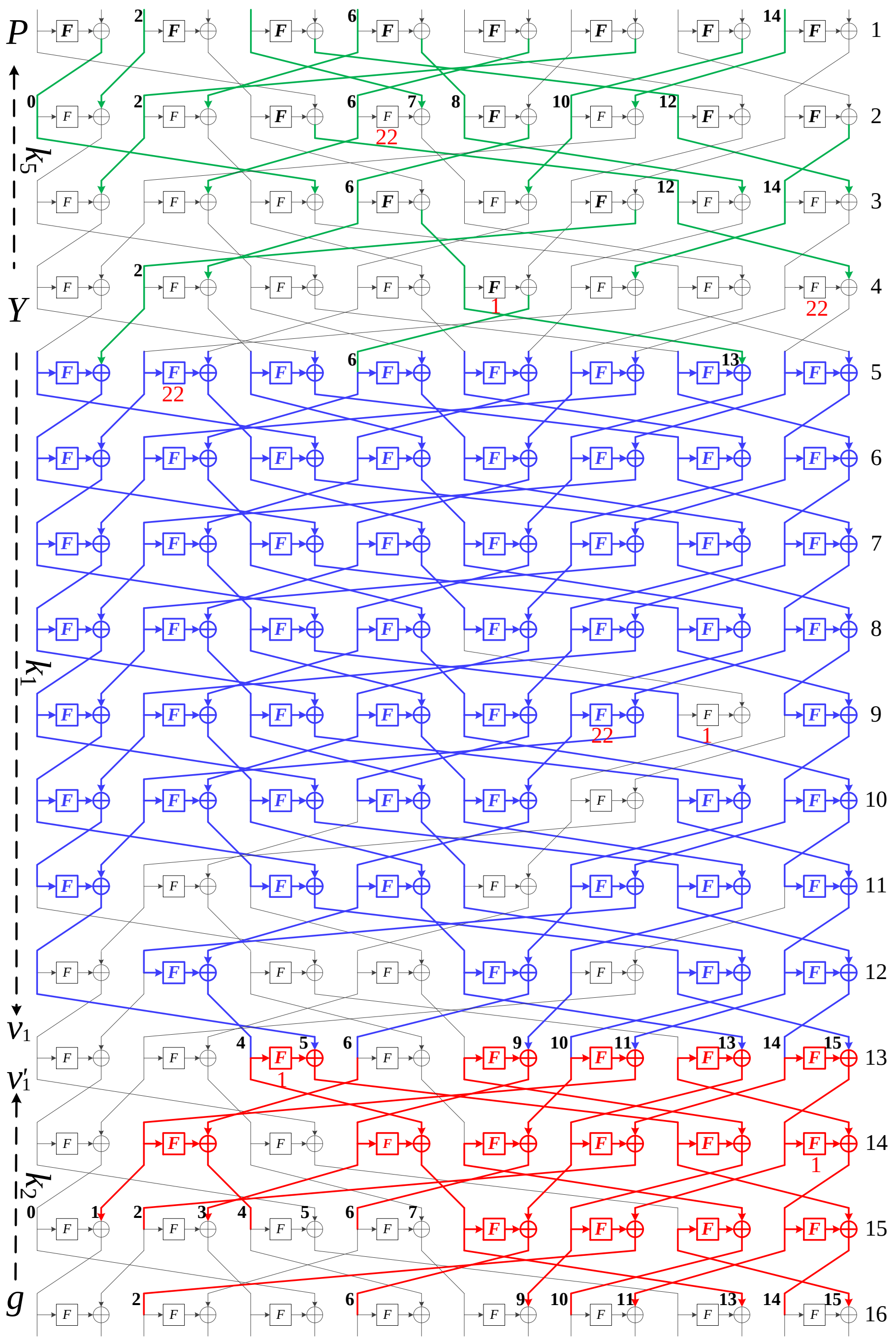

Fig. 4: round 1 to round 16 of Boztaş *et al.*'s attack on 25-round TWINE-128

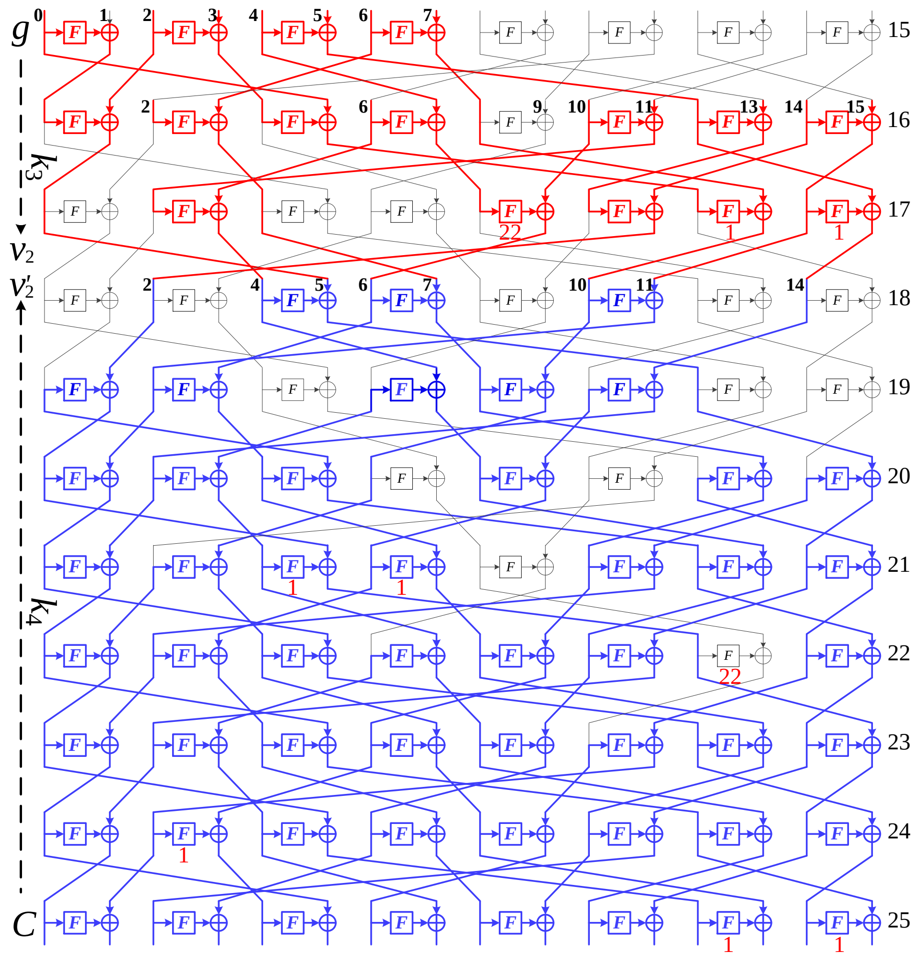

Fig. 5: round 15 to round 25 of Boztaş *et al.*'s attack on 25-round TWINE-128

2. Decrypt Y without guessing κ_8^{22} and get the plaintext value. Get the corresponding ciphertext value and continue to decrypt to get the value of v_2'. Construct T_2 containing all values of k_4 indexed by $v_2||k_{c_2}$.
3. Choose a fixed value of g
   (a) Do a MITM attack between Y and g and write possible keys (k_1, k_2) to a table Q.
   (b) Do a MITM attack between g and C and check whether possible keys (k_3, k_4) are also in Q. If true, then output (k_1, k_2, k_3, k_4) as a right key candidate. Test these values with other PC pairs.
   (c) If no right key found, change the value of g.

The dominant time complexity claimed in [4] lies in Step 3.(a) and Step 3.(b), and both of them are computed as $2^{64+60} \cdot 2.5/25 \approx 2^{121}$ 25-round encryptions.

The time complexity of Step 3.(b) is about 2^{121} encryptions because X_{16}^{9} can be fixed in Step 3.(b). Then, the total time complexity is about 2^{122} 25-round encryptions. After a closer look at Step 3.(a), we find that $X_{15}^{0,5,7}$ can actually be fixed in Step 3.(a). This observation will help to reduce the time complexity of Step 3.(a).

3 Revisiting Time Complexity of 2D-MITM Attack on 25-Round TWINE-128

In this section, we rewrite 2D-MITM attack on 25-round TWINE-128 (with improved Step 3 considering the above observation) in an algorithmic form (Algorithm 1) and show that the actual time complexity of the attack exceeds the exhaustive search. Algorithm 1 generally follows Boztaş *et al.*'s attack procedure while we integrate the above observation into the procedure to help reduce the time complexity of Step 3.(a) of their attack on 25-round TWINE-128. Some comments are added to help understanding the attack procedure.

As wrote earlier, in [4] the authors claim that the dominant part of time complexity of the attack on 25-round TWINE-128 lies in Step 3.(a) and Step 3.(b), which is about 2^{121} 25-round encryptions for each step. However, Step 1 (line 1 to line 3) and Step 2 (line 4 to line 6) actually have comparable time complexity. E_{f_1} contains about 7 rounds of encryptions and computing v_2' from Y in Step 2 (line 4 to line 6) needs about 8 decryptions. Thus, the time complexity of Step 1 and Step 2 is about $2^{124} \cdot (7/25 + 8/25) \approx 2^{123.3}$ 25-round encryptions. From line 8 to line 12 of Algorithm 1, we show that the time complexity of Step 3.(a) can be reduced because v_1' has nothing to do with $X_{15}^{0,5,7}$. In the way presented in Algorithm 1, the time complexity of Step 3.(a) is about $2^{48} \cdot 2^{4} \cdot 2^{60} \cdot 2.5/25 \approx 2^{108.7}$ 25-round encryptions, which can be negligible compared with Step 1 and Step 2. The side effect of the new way to proceed Step 3.(a) is that after line 12, table Q contains more (k_1, k_2) pairs. Since the memory requirements are dominated by Step 1 and Step 2, this side effect is trivial. Step 3.(b) in Algorithm 1 still have the time complexity of about 2^{121} 25-round encryptions. Thus, about $2^{123.3} + 2^{121} \approx 2^{123.6}$ 25-round encryptions are preformed and the attack on 25-round TWINE-128 in [4] seems alright.

Yet, encryptions and decryptions are not the only operations that are performed in the attack. There are a few operations that seem can be done in one operation at the first glance requiring massive operations actually. Pay attention to line 12 of Algorithm 1, as each entity in T_1 contains 2^{40} k_1, adding (k_1, k_2) to Q is actually adding 2^{40} (k_1, k_2) values to Q. Can this operation be done in one operation? We don't know for sure. Next is about line 17 of Algorithm 1. Each entity of T_2 contains 2^{44} k_4, so check whether $(k_3, k_4) \in Q$ need to be repeated for all k_4, namely 2^{44} times. Thus this check operation is performed about $2^{48} \cdot 2^{12} \cdot 2^{64} \cdot 2^{44} = 2^{168}$ times. The situation becomes even worse, if we consider that we are actually checking whether a value exists in a table of size 2^{104}. Maybe Q could be constructed in a more smarter way and the attacker could determine whether a value exists in Q instantly (although to the best

Algorithm 1: Boztaş *et al.*'s attack with improved Step 3

```
   // Step 1 from line 1 to line 3
1  for all values of 124-bit $k_1$ do
2      Compute $v_1 = E_{f_1}(k_1, Y)$
3      Add $k_1$ to $T_1[v_1||k_{c_1}]$
   // $T_1$ contains $2^{124}$ $k_1$, each entity contains $2^{40}$ $k_1$
   // Step 2 from line 4 to line 6
4  for all values of 124-bit $k_4$ do
5      Compute $v'_2$ from $Y$ // $P = E^{-1}_{b_3}(k_5, Y)$, ask for $C$, $v'_2 = E^{-1}_{b_2}(k_4, C)$
6      Add $k_2$ to $T_2[v'_2||k_{c_2}]$
   // $T_2$ contains $2^{124}$ $k_4$, each entity contains $2^{44}$ $k_4$
   // Step 3 from line 7 to line 21
7  for all values of 48-bit $X^{1,2,3,4,6}_{15}||X^{2,6,20,11,13,14,15}_{16}$ do
       // Step 3.(a) from line 8 to line 12
8      for all values of 4-bit $X^9_{16}$ do
9          for all values of 60-bit $k_2$ do
10             Compute $v'_1 = E^{-1}_{b_1}(k_2, g)$ // no relation between $v'_1$ and $X^{0,5,7}_{15}$
11             Extract the entity with index $v'_1||k_{c_1}$ from $T_1$
12             Add $(k_1, k_2), k_1 \in T_1[v'_1||k_{c_1}]$ to a table $Q$ // one operation?
       // $Q$ contains $2^4 \cdot 2^{60} \cdot 2^{40} = 2^{104}$ $(k_1, k_2)$
       // Step 3.(b) from line 13 to line 21
13     for all values of 12-bit $X^{0,5,7}_{15}$ do
14         for all values of 64-bit $k_3$ do
15             Compute $v_2 = E_{f_2}(k_3, g)$// no relation between $v_2$ and $X^9_{16}$
16             Extract the entity with index $v_2||k_{c_2}$ from $T_2$
17             Check whether $(k_3, k_4), k_4 \in T_2[v_2||k_{c_2}]$ is in $Q$ // Caution!
18             if find one match of $(k_1, k_2)$ and $(k_3, k_4)$ then
19                 Exhaustively test $(k_1, k_2, k_3, k_4)$ with a few $PC$ pairs
20                 if key tuple $(k_1, k_2, k_3, k_4)$ passes all tests then
21                     return $(k_1, k_2, k_3, k_4)$ as the right key
```

of our knowledge, there is no such way to achieve this goal). Nevertheless, the checking operation in line 17 will always be performed far more than 2^{128} times and the time complexity of line 17 actually exceeds the exhaustive search. Thus, the whole 2D-MITM attack on 25-round TWINE-128 exceeds exhaustive search.

References

1. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A Lightweight Block Cipher for Multiple Platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339-354. Springer, Heidelberg (2013)

2. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
3. Zhu, B., Gong, G.: Multidimensional Meet-in-the-Middle Attack and Its Applications to KATAN32/48/64. IACR Cryptology ePrint Archive, 2011:619 (2011)
4. Boztaş, Ö., Karakoç, F., Çoban, M.: Multidimensional Meet-in-the-Middle Attacks on Reduced-Round TWINE-128. In: Avoine, G., Kara, O. (eds.) LIGHTSEC 2013. LNCS, vol. 8162, pp. 55–67. Springer, Heidelberg (2013)