Generalized MitM Attacks on Full TWINE

Mohamed Tolba, Amr M. Youssef*

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Québec, Canada.

Abstract

TWINE is a lightweight block cipher which employs a generalized Feistel structure with 16 nibble-blocks. It has two versions: TWINE-80 and TWINE-128, both have a block length of 64 bits and employ keys of length 80 and 128 bits, respectively. In this paper, we propose a low data complexity key recovery attack on the full cipher. This attack is inspired by the 3-subset Meet-in-the-Middle (MitM) attack. However, in our attack, we remove the restrictions of the 3-subset MitM by allowing the key to be partitioned into $n \geq 3$ subsets and by not restricting these subsets to be independent. To improve the computational complexity of the attack, we adopt a recomputation strategy similar to the one used in the original biclique attack. Adopting this approach, we present a known plaintext key recovery attack on TWINE-80 and TWINE-128 with time complexities of $2^{78.74}$ and $2^{126.1}$, respectively. Both attacks require only two plaintext-ciphertext pairs. Furthermore, by combining our technique with a splice-and-cut approach, we gain a slight improvement in the time complexity of the attack at the expense of increasing the number of required plaintext-ciphertext pairs.

Keywords: Cryptanalysis, Meet-in-the-Middle, Low data complexity attacks, TWINE, Bicliques

1. Introduction

Recently, there has been a rapid increase in utilizing resource constrained devices such as wireless sensor networks and RFIDs. The limited resources (e.g., memory, battery life and processing power) available on these devices impose challenging requirements on the cryptographic primitives that can be deployed on them. Over the past few years, several new lightweight block ciphers were proposed (e.g., PRESENT [5], KATAN/KTANTAN [8], Zorro [11], HIGHT [12], and TWINE [15] [16]). These ciphers use new design concepts that aim to reduce the algorithm footprints on resource constrained devices. In particular, the majority of these lightweight ciphers tend to employ simple key schedules with relatively slow diffusion. Therefore, it seems intuitive to develop advanced variants of the basic MitM attack in order to evaluate the security margins of these ciphers. At the ECRYPT workshop on lightweight cryptography, TWINE was proposed by Suzaki *et al* [15].

*Corresponding author
Email address: youssef@ciise.concordia.ca (Amr M. Youssef)

Afterwards, it was presented at SAC 2012 [16]. TWINE adopts a generalized Feistel structure with 16 nibble-blocks. It has two versions: TWINE-80 and TWINE-128, both with a block length of 64 bits, iterate over 36 rounds, and employ keys of length 80 and 128 bits, respectively. While the encryption/decryption structure of both variants of the cipher are identical, each variant has its own key scheduling algorithm.

In this paper, we present a low data complexity attack on both TWINE-80 and TWINE-128 in the single key attack model. Our attack can be seen as a generalization of the 3-subset MitM attack that partitions the key into 3 disjoint subsets [6]. In this generalized attack, we allow the key to be partitioned into $n \geq 3$ subsets and we do not restrict these subsets to be independent. These n subsets are used in the forward and backward computations of the MitM attack, and the effect of dependency is handled by a re-computation technique in a manner similar to the recomputation phase of the biclique cryptanalysis [3] [4] [17]. Using this approach, we present a known plaintext key recovery attack on TWINE-80 and TWINE-128 with time complexities of $2^{78.74}$ and $2^{126.1}$, respectively. Both attacks require only two plaintext-ciphertext pairs, which is equal to the unicity distance of the cipher. Furthermore, by combining our technique with a splice-and-cut approach, we can gain a slight improvement in the time complexity of the attack at the expense of increasing the number of required plaintext-ciphertext pairs. In particular, using 2^{32} chosen plaintext-ciphertext pairs, the time complexity is reduced to $2^{78.63}$ and $2^{125.97}$ for TWINE-80 and TWINE-128, respectively.

Table 1: Summary of the current cryptanalysis results on TWINE-80 (KP: Known Plaintext. CP: Chosen Plaintext)

Attack	# Rounds	Time	Data	Memory	Reference
Bilcique	36	$2^{79.1}$	2^{60}CP	2^{8}	[9]
Generalized MitM	36	$2^{78.74}$	2KP	2^{9}	Section 4
Generalized MitM with splice-and-cut	36	$2^{78.63}$	2^{32}CP	2^{9}	Section 5

Table 2: Summary of the current cryptanalysis results on TWINE-128 (KP: Known Plaintext. CP: Chosen Plaintext)

Attack	# Rounds	Time	Data	Memory	Reference
Biclique	36	$2^{126.82}$	2^{60}CP	2^{8}	[9]
Generalized MitM	36	$2^{126.1}$	2KP	2^{9}	Section 4
Generalized MitM with splice-and-cut	36	$2^{125.97}$	2^{32}CP	2^{9}	Section 5

It should be noted that biclique cryptanalysis of TWINE was presented in [9] [13] and a multidimensional MitM attack was presented in [7] (see also [18]). Recently, Biryukov *et al.* presented a MitM attack on TWINE-128 reduced to 25 rounds. The attack data, time and memory complexities are given by $2^{48}, 2^{124.7}$ and 2^{109}, respectively [2]. The only attack that considers the full versions of TWINE-80 and TWINE-128 is the biclique cryptanalysis in [9] with time complexity $2^{79.1}$, and $2^{126.82}$, respectively. The data complexity of this attack is 2^{60} for the two variants of TWINE which is clearly not practical given the nature of the lightweight environment in which these ciphers are likely to be deployed. Tables 1 and 2 contrast our results with the previous cryptanalytic results, in the single key model, on the full versions of TWINE-80 and TWINE-128, respectively.

The rest of the paper is organized as follows. Section 2 presents an overview of the original 3-subset MitM attack. In section 3, we provide the notation used throughout the rest of this paper and a brief

description of TWINE. Our attack is presented in section 4. The combination of our new attack with the splice-and-cut attack is presented in section 5. Finally, we conclude our work in section 6.

2. An overview of the three-subset Meet-in-the-Middle attack

A 3-subset MitM attack [6] is a generalization of the basic MitM which was originally proposed by Diffie and Helman [10]. The two main stages of this attack are:

1. MitM stage: which is responsible for filtering out some wrong key candidates, thereby reducing the remaining key search space.
2. Key testing stage: which is responsible for finding the right key among the remaining key candidates in a brute force manner.

Let $E_K : \{0,1\}^b \rightarrow \{0,1\}^b$ be an r rounds block cipher with b-bit block length, and k-bit key. E_K can be decomposed into two sub-ciphers as follows:

$$E_K = G_{K_2} \circ F_{K_1}(x), x \in \{0,1\}^b,$$

where F_{K_1} is a sub-cipher that uses a set of key bits, K_1, and G_{K_2} is a sub-cipher that uses a set of key bits, K_2. Let $A_0 = K_1 \cap K_2$ be the set of common key bits that are used in the two sub-ciphers and let $A_1 = K_1 \backslash K_1 \cap K_2$ and $A_2 = K_2 \backslash K_1 \cap K_2$ denote the set of key bits that are used only in F_{K_1} and G_{K_2}, respectively. The MitM stage, which requires only one plaintext-ciphertext pair, can be summarized as follows:

- Create an empty table t_{cnd} which will contain the surviving key candidates after this stage.
- For each key value in A_0
  - Create an empty table t_{aux}.
  - Compute $v = F_{K_1}(P)$ for each key value in A_1, and add a new entry (v, A_1) to the table t_{aux} indexed by v.
  - Compute $u = G^{-1}_{K_2}(C)$ for each key value in A_2, and search for u in t_{aux}. If exists, add a new entry (K_1, K_2) to t_{cnd}.

Let $|A_i|$ denote the size of A_i in bits. Then, after this stage, the number of surviving keys $\approx (2^{|A_1|+|A_2|}/2^b) \times 2^{|A_0|} = 2^{k-b}$. In the key testing stage, the remaining key candidates are tested in an exhaustive manner using additional plaintext-ciphertext pairs. Uniquely determining the right key requires about $\lceil l/b \rceil$ known plaintext-ciphertext pairs, where $l = k - m$, and m is the length of the matching variable in bits. Thus, the time complexity of the attack is given by:

$$T_c = \overbrace{2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})}^{\text{MitM stage}} + \overbrace{(2^{k-b} + 2^{k-2b} + ...)}^{\text{Key testing stage}}$$

To gain a computational advantage over exhaustive search, both A_1 and A_2 should not be empty. The data complexity of this attack ($\approx \lceil l/b \rceil$) is dominated by the data required in the key testing stage and the memory complexity is determined by the matching step which requires saving one of the two sets, i.e., A_1 or A_2. Therefore, the memory complexity is given by $min(2^{|A1|}, 2^{|A_2|})$.

3. Specifications of TWINE

The following notation will be used throughout the rest of the paper:

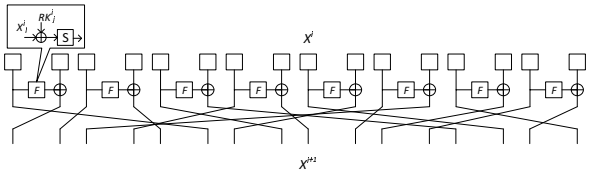

Figure 1: The TWINE round function

- K: The master key.
- RK^i: The 32-bit key used in round i.
- K^i: The 80 or 128 bits generated from K after i rounds to obtain the round key RK^i.
- $K^i[j]$: j^{th} nibble of K^i. The indices of the nibbles begin from 0.
- $K^i[i, j, \cdots, l]$: i^{th}, j^{th}, $\cdots$, and l^{th} nibbles of K^i.
- X^i: The 16 4-bit nibbles output of round i.
- $X^i[j]$: j^{th} nibble of X^i.
- $X^i[i, j, \cdots, l]$: i^{th}, j^{th}, $\cdots$, and l^{th} nibbles of X^i.

3.1. Specifications of TWINE

Both TWINE-80 and TWINE-128 deploy the same generalized Feistel structure where the only difference between them is the key schedule. The encryption/decryption operations in TWINE iterate over 36 rounds where, as depicted in Figure 1, each round consists of three operations: key addition, 4-bit s-box lookups, and a permutation operation.

The key schedule of TWINE-80 produces 36 round keys of length 32-bit from the 80-bit master key, K, as follows:

- $K^1 = K$
- $RK^1 = K^1[1, 3, 4, 6, 13, 14, 15, 16]$
- for $i = 2, 3, ..., 36$
  - $K^i = K^{i-1}$
  - $K^i[1] = K^i[1] \oplus S(K^i[0])$
  - $K^i[4] = K^i[4] \oplus S(K^i[16])$
  - $K^i[7] = K^i[7] \oplus (0||CON^i_H)$
  - $K^i[19] = K^i[19] \oplus (0||CON^i_L)$
  - $K^i[0, 1, 2, 3] = K^i[0, 1, 2, 3] \lll 4$
  - $K^i = K^i \lll 16$
  - $RK^i = K^i[1, 3, 4, 6, 13, 14, 15, 16]$

Similarly, the key schedule for TWINE-128 produces 36 round keys of length 32-bit from the 128-bit master key, K, as follows:

- $K^1 = K$
- $RK^1 = K^1[2, 3, 12, 15, 17, 18, 28, 31]$
- for $i = 2, 3, ..., 36$
  - $K^i = K^{i-1}$
  - $K^i[1] = K^i[1] \oplus S(K^i[0])$
  - $K^i[4] = K^i[4] \oplus S(K^i[16])$
  - $K^i[23] = K^i[23] \oplus S(K^i[30])$
  - $K^i[7] = K^i[7] \oplus (0||CON^i_H)$
  - $K^i[19] = K^i[19] \oplus (0||CON^i_L)$
  - $K^i[0, 1, 2, 3] = K^i[0, 1, 2, 3] \lll 4$
  - $K^i = K^i \lll 16$
  - $RK^i = K^i[2, 3, 12, 15, 17, 18, 28, 31]$

where S is a 4×4 s-box and CON^i_H, CON^i_L are predefined constants. For further details, the reader is referred to [15].

4. Proposed Attack

In order to apply the 3-subset MitM attack, one needs to find two independent subsets, A_1 and A_2, from the master key bits. Finding these two subsets is hard and possibly no two independent subsets can cover the whole cipher. We solve this problem by relaxing this condition. More precisely, in our attack, the $n \geq 3$ subsets produced by the key partitioning process are allowed to be dependent. However, using these dependent subsets raises a new problem of how to efficiently compute the inner state nibbles that depend on more than one subset. The re-computation technique utilized in the biclique cryptanalysis [4] provides a natural solution to this problem; we compute and store the nibbles that depend on each subset, then nibbles that depend on more than one subsets are recomputed.

Relaxing this dependency restriction and using n partitions for the key (which produces n subsets, $A_1, A_2, \cdots, A_n$, in addition to the common subset A_0) enable us to better minimize the computational complexity of the attack. Moreover, there are many block ciphers with invertible r rounds key schedule where knowing an intermediate key $K^i, i = 1, 2, .., r$, allows us to recover the master key K. Hence, guessing an intermediate key $K^i, i = 2, 3, ..., n-1$, and running the key schedule in both directions allow us to better utilize the slow diffusion of the key schedule to extend the number of attacked rounds [7].

Complexity analysis:. To calculate the computational complexity of this attack, we need to count the number of s-box lookups which correspond to the most consuming time operations in the encryption/decryption process. Similar to the 3-subset MitM attack, our attack also consists of MitM with a re-computation stage, and a key testing stage. Suppose that the key K is partitioned into n subsets where each subset contains d bits of the master key. Also assume that the total number of s-boxes used in the full cipher and the key schedule is SS. The complexity of the s-boxes lookup operations during the re-computation stage is given by

$$T_{c_s} = \left(\frac{\sum_{i=0}^{n} 2^{i \times d} \times S_i}{SS} \right),$$

where S_i is the number of s-boxes that depend on i subsets, $i = 0, 1, 2, ..., n$. For example, if $n = 3$, then S_2 denotes the number of s-boxes that depend on A_1, and A_2; A_1, and A_3; and A_2, and A_3. Let T_{c_L} denote the computational complexity associated with accessing the pre-computation tables during the re-computation stage. Thus, the total computational complexity of our attack is given by

$$T_c = 2^{|K|-nd} \Bigg(\overbrace{T_{c_s} + T_{c_L}}^{\text{Re-computation stage}} + \overbrace{2^{nd-m}}^{\text{Key testing stage}} \Bigg)$$

where $|K| - nd$ is the length of the common set A_0, and m is the length of the matching variable in bits. The data complexity D_c of the attack is determined by the unicity distance of the cipher $= \lceil l/b \rceil$ and the memory complexity is given by $n \times 2^d$, which corresponds to the memory required to store each subset computation so that it can be used later in the re-computation. The computational complexity of the attack T_c, for a given n, is optimized over the following parameters:

1. Key index (KI): This parameter indicates which K^i is to be partitioned into n subsets.

2. Matching round (MR): This parameter indicates the round at which the matching takes place (i.e., we do the matching at the end of round MR.)
3. Matching nibbles (MN): This parameter indicates which nibbles (from the 16 inner state nibbles) are used for matching.

Since in TWINE, we have $\binom{16}{i}$ for $i = 1, 2, \cdots, 16$, options to choose nibbles for matching, we restricted our search program to perform partial matching on one or two nibbles. Also, since in TWINE, all the operations are performed on the nibbles level, we restrict our search program to perform the partition on the nibble level as well.

4.1. Low data complexity attack on TWINE-80

In this section, we illustrate the application of the proposed attack on the full round TWINE-80. The optimized attack parameters are obtained by performing full partitioning of the key, i.e., setting $n = 20$, and then using exhaustive search to optimize the time complexity of the attack over all possible values of the remaining parameters, i.e., KI, MR, and MN. The resulting attack requires only two plaintext-ciphertext pairs, its time complexity is $2^{78.74}$, and its memory complexity is 2^9.

The parameters used in this attack are:

- $A_i = K^{20}[i-1]$ for $i = 1, 2, ..., 20$ and $A_0 = \phi$.
- $KI = K^{20}$, $MR = 10$ and $MN = X^{10}[0, 10]$.

Since it is difficult to visualize the attack in this case, we choose to visualize the attack for a simpler case when $n = 3$. For this case, the best attack found by our search corresponds to $KI = K^{26}$,

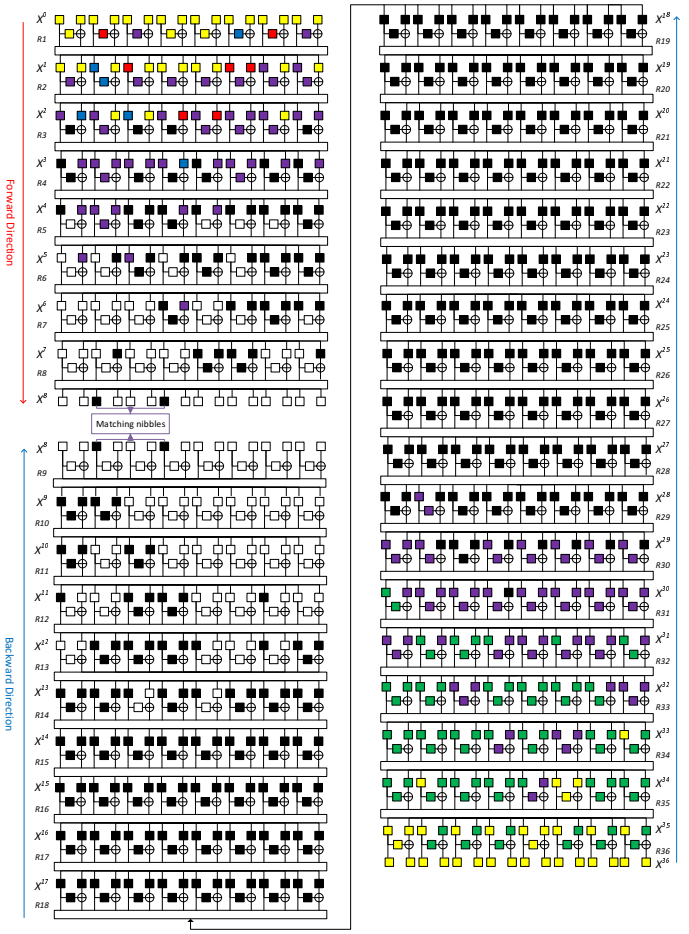

Figure 2: Generalized Meet-in-the-Middle attack on full TWINE-80 for $n = 3$

$A_1 = K^{26}[11]$, $A_2 = K^{26}[16]$, $A_3 = K^{26}[19]$, $A_0 = K^{26} \setminus \{A_1, A_2, A_3\}$, $MR = 8$ and $MN = X^8[2, 6]$. Figures 2 and 3 depict this generalized MitM attack on full TWINE-80 and the key schedule dependency, respectively.

The computational complexity of the attack is calculated by counting the number of s-boxes needed for computing the matching variables v and u. This number is then normalized by dividing it by the number of s-boxes used in 36-rounds of TWINE-80 (358 s-boxes) to determine how many encryption operations are needed. The following color notation

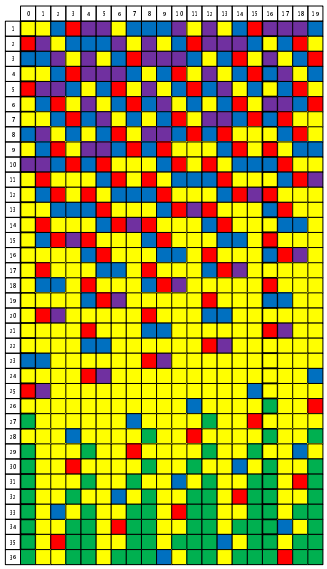

Figure 3: Key schedule dependency for TWINE-80

is used in Figures 2 and 3:

- Yellow: denotes the nibbles that are not affected by changing any of the key subsets A_1, A_2 and A_3 but are effected by changing the common subset A_0. Thus, these nibbles are computed once for each value of A_0.
- Blue, green and red: denote the nibbles that are affected by changing either A_1, A_2 or A_3, respectively. Thus, these nibbles are computed 2^4 times for every value of A_0.
- Purple: denotes the nibbles that are affected by changes in any combination of the two key subsets A_i and A_j. Thus, these nibbles are computed 2^8 times for every value of A_0.
- Black: denotes the nibbles that are affected by changes in the three key subsets A_1, A_2 and A_3. Thus, these nibbles are computed 2^{12} times for every value of A_0.
- White: denotes the nibbles that do not affect the calculation of the matching nibbles and, hence, they do not need to be calculated.

The aim of the MitM with re-computation stage is to filter the key space. Any key candidate is rejected if the condition $\overrightarrow{X^8[2,6]} = \overleftarrow{X^8[2,6]}$ does not hold.

The forward computation required to calculate $\overrightarrow{X^8[2,6]}$ from P proceeds as shown in Algorithm 1.

The same strategy can be applied to the backward computation in order to calculate $\overleftarrow{X^8[2,6]}$ from C.

The number of s-boxes that are calculated in the cipher and the key schedule in the forward and backward directions for each value of A0 is given by: 3 (for the forward computation)+3 (for the backward computation) + 33 for the key schedule =39 s-boxes which are computed once, 4+28+33= 65 s-boxes which are computed 2^4 times, 16+25+4= 45 s-boxes which are computed 2^8 times and 21+136+0=157 s-boxes which are computed 2^{12} times.

In the key testing stage, the number of surviving candidates for each value of A_0 is given by $2^{12-8} = 2^4$ since we have 2^{12} keys in each A_0 and we match on 8 bits. Therefore, we have $2^{68+4} = 2^{72}$ surviving candidates for all the values of A_0. These surviving candidates need to be tested using $\lceil l/b \rceil = \lceil 72/64 \rceil = 2$ additional plaintext-ciphertext pairs to uniquely determine the key.

In each iteration of the first *for* loop in Algorithm 1, we need to access t_i for 15 times to store the re-computed nibbles. Then, in each iteration of the second *for* loop in Algorithm 1, we need to access

Algorithm 1: Forward computation for n=3

forall the *values of* A_0 **do**
- /* Perform the base computation */
- Partially encrypt P with the bits of $A_1, A_2, and A_3$ set to zero to obtain $\overrightarrow{X^8[2,6]}$;
- Store all the intermediate (base) computation;
- **for** $i = 1, 2, 3$ **do**
  - /* Iteration $i = 1$ computes the blue nibbles; and the purple and black nibbles that are affected by changing A_1 */
  - /* Iteration $i = 2$ computes the green nibbles; and the purple and black nibbles that are affected by changing A_2 */
  - /* Iteration $i = 3$ computes the red nibbles; and the purple and black nibbles that are affected by changing A_3 */
  - /* The total memory requirement for all the iterations in this loop is 3×2^4 */
  - **forall the** *non-zero values in* A_i **do**
    - Re-compute the nibbles that are affected by changes in A_i (while the bits corresponding to the other two sets are set to zero);
    - Store the recomputed nibbles in table t_i (t_i contains 2^4 forward computation);
- **for** $i = 1, 2$ **do**
  - **for** $j = i + 1$ *to* 3 **do**
    - /* Each iteration re-computes purple nibbles and black nibbles that are affected by A_i and A_j */
    - **forall the** *non-zero values in* A_i, *and* A_j **do**
      - Re-compute the nibbles that are affected by changes in both A_i and A_j only using the pre-computed values in t_i and t_j;
- **forall the** *non-zero values in* A_1, A_2, *and* A_3 **do**
  - Re-compute the nibbles that are affected by changes in A_1, A_2 and A_3 (i.e., the black nibbles) using the pre-computed values in t_1, t_2 and t_3;

t_i for 15 times and t_j for 15^2 times. Finally, in the last *for* loop of Algorithm 1, we need to access t_1, t_2 and t_3 for 15, 15^2 and 15^3 times, respectively. Assuming that the time complexity of accessing these lookup tables is the same as s-box lookup complexity (both requires an index of size 4 bits), then the computation complexity associated with accessing these pre-computation tables in the MitM stage is given by [1]

$$2^{68}\left(\frac{15\times 3+((15+15^2)\times 3)+(15+15^2+15^3)}{358}\right) \approx 2^{71.61},$$

The s-boxes lookup complexity in the MitM stage is given by

$$2^{68}\left(\frac{39+(2^4\times 65)+(2^8\times 45)+(2^{12}\times 157)}{358}\right)\approx 2^{78.84}.$$

Thus the total complexity of the attack is given by

$$2^{71.61}+2^{68}\left(\frac{39+(2^4\times 65)+(2^8\times 45)+(2^{12}\times 157)}{358}+2^4\right) \approx 2^{78.85}.$$

Note that we need a computation on purple nibbles if we have a change in one of yellow, blue, green, or red nibbles. The term 2^8 in the above equation already accounts for these cases, i.e., for the case where one of A_i and A_j is set to zero (a change in blue, green or red) and the case where both of them are set to zero (a change in yellow). Similarly, the term 2^{12} also accounts for the cases where all the A_i sets are non-zeros in addition to the cases where a subset of them is zero.

The memory complexity is upper bounded by

[1] In general, for n partitions, the total complexity associated with accessing the pre-computation tables is given by

$$2^{|K|-nd}\times\left(\frac{\sum_{i=1}^{n}\binom{n}{i}\sum_{j=1}^{i}(2^d-1)^j}{SS}\right).$$

3×2^4 full TWINE-80 computations, and the data complexity is two plaintext-ciphertext pairs.

4.2. A low data complexity attack on TWINE-128

The only difference between TWINE-80 and TWINE-128 is the key schedule. For TWINE-128, the best attack complexity is achieved when we made a full partitioning to the key, i.e., by setting $n = 32$. This attack requires $2^{126.1}$ full round TWINE-128 encryption operations, its memory complexity is upper bounded by $32 \times 2^4 = 2^9$, and its data complexity is two plaintext-ciphertext pairs. The parameters obtained by our search program are:

- $A_i = K^{19}[i-1]$ for $i = 1, 2, ..., 32$ and $A_0 = \phi$.
- $KI = K^{19}$, $MR = 13$, and $MN = X^{13}[2, 6]$.

5. A generalized MitM attack on TWINE with splice-and-cut

Several improvements to the basic 3-subset MitM attack have been presented. One of these improvements is the splice-and-cut technique that was proposed by Aoki and Sasaki [1] to present a preimage attack on the SHA-0 and SHA-1 hash functions. As depicted in Figure 4, this technique differs from the basic 3-subset MitM attack in that the beginning of the forward/backward directions is not restricted to the plaintext/ciphertext. Instead, we may choose any intermediate variable X, partially decrypt X to obtain the corresponding plaintext P, and using the encryption oracle we get its corresponding ciphertext C. Then, partially decrypt C to get the matching variable u. Afterwards, we partially encrypt X to obtain v. The data complexity of this attack is determined by the bits of the plaintext that are affected when using K_2 to partially decrypt X. It should be noted that this attack changes the nature of the basic 3-subset MitM from a known plaintext to a chosen plaintext attack.

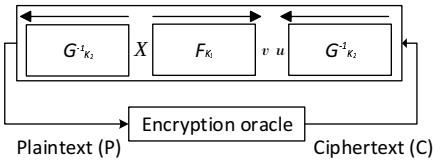

Figure 4: Meet-in-the-Middle with splice-and-cut technique

In what follows, we combine the attack presented in the previous sections with this splice-and-cut technique. The combined attack leads to an improvement in the time complexity at the expense of increasing the data complexity. Using the splice-and-cut technique adds one more parameter XP to our optimization problem. When $XP = i$, it means that X is chosen to be the input to round i.

For TWINE-80, the parameters used in our attack that utilizes full key partitioning are given by:

- $A_i = K^{20}[i-1]$ for $i = 1, 2, ..., 20$ and $A_0 = \phi$.
- $KI = K^{20}$, $MR = 11$, $MN = X^{11}[2, 6]$ and $XP = 2$.

The previous parameters allow a key recovery attack with computational complexity of $2^{78.63}$, memory complexity of 2^9 and data complexity of 2^{32} chosen plaintext.

The corresponding parameters for our attack on TWINE-128 are given as follows:

- $A_i = K^{20}[i-1]$ for $i = 1, 2, ..., 20$ and $A_0 = \phi$.
- $KI = K^{20}$, $MR = 14$, $MN = X^{14}[2, 6]$, and $XP = 2$.

Using these parameters allows a key recovery attack with computational complexity of $2^{125.97}$, memory complexity of 2^9 and data complexity of 2^{32} chosen plaintext.

6. Conclusion

We presented a low data complexity key recovery attack on TWINE-80 and TWINE-128. Our attack generalizes the original 3-subset MitM attack by allowing the key to be partition into, possibly dependent, $n \geq 3$ subsets. It also utilizes some ideas from the recomputation phase of the biclique cryptanalysis to reduce the time complexity of the attack. Combining this attack with the splice-and-cut technique allows for some data-time trade off. To the best of our knowledge, both proposed attacks present the best attacks on the full TWINE-80 and TWINE-128 with respect to both the time and data complexities.

The idea behind the proposed attack is general enough and can be applied to other lightweight ciphers as well. Meanwhile, similar to the biclique cryptanalysis, our attack can be described as a bruteforce-like cryptanalysis [14] which is not able to conclude that a particular primitive has some cryptanalytic weakness. However it can help to better understand the real security provided by the primitive when no attack tweaks are adopted. While most of the applications of bruteforce-like cryptanalysis have an advantage that is sometimes much smaller than a factor of 2, for lightweight ciphers with key sizes of 80 bits or less, this can be very useful to know. To this end, designers of lightweight symmetric primitives should consider this class of attacks during the assessment of new designs.

7. Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions that helped improve the quality of the paper. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant N00930.

8. References

[1] Aoki, K., and Sasaki, Y. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *CRYPTO* (2009), S. Halevi, Ed., vol. 5677 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 70–89.

[2] Biryukov, A., Derbez, P., and Perrin, L. P. Differential Analysis and Meet-in-the-Middle Attack against Round-Reduced TWINE. In *Fast Software Encryption 2015*, G. Leander, Ed., vol. 9054 of *Lecture Notes in Computer Science*, Springer, pp. 3–27.

[3] Bogdanov, A., Chang, D., Ghosh, M., and Sanadhya, S. Bicliques with minimal data and time complexity for AES. In *International Conference on Information Security and Cryptology (ICISC 2014)*, J. Lee and J. Kim, Eds., vol. 8949 of *Lecture Notes in Computer Science*, Springer, pp. 160–174.

[4] Bogdanov, A., Khovratovich, D., and Rechberger, C. Biclique cryptanalysis of the full AES. In *Proceedings of the 17th International Conference on The Theory and Application of Cryptology and Information Security* (2011), ASIACRYPT'11, Springer-Verlag, pp. 344–371.

[5] Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J., Seurin, Y., and Vikkelsoe, C. PRESENT: an ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007* (Berlin, Heidelberg, 2007), P. Pail-

lier and I. Verbauwhede, Eds., CHES '07, Springer-Verlag, pp. 450–466.

[6] Bogdanov, A., and Rechberger, C. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *Selected Areas in Cryptography* (2010), A. Biryukov, G. Gong, and D. R. Stinson, Eds., vol. 6544 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 229–240.

[7] Boztaş, O., Karakoç, F., and Çoban, M. Multidimensional meet-in-the-middle attacks on reduced-round TWINE-128. In *Lightweight Cryptography for Security and Privacy* (2013), G. Avoine and O. Kara, Eds., vol. 8162 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, pp. 55–67.

[8] Cannière, C., Dunkelman, O., and Knežević, M. KATAN and KTANTAN - A family of small and efficient hardware-oriented block ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009* (2009), C. Clavier and K. Gaj, Eds., vol. 5747 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 272–288.

[9] Çoban, M., Karakoç, F., and Özkan Boztaş. Biclique cryptanalysis of TWINE. In *Cryptology and Network Security* (2012), J. Pieprzyk, A.-R. Sadeghi, and M. Manulis, Eds., vol. 7712 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 43–55.

[10] Diffie, W., and Hellman, M. E. Special feature exhaustive cryptanalysis of the NBS data encryption standard. *Computer 10*, 6 (June 1977), 74–84.

[11] Gèrard, B., Grosso, V., Naya-Plasencia, M., and Standaert, F.-X. Block ciphers that are easier to mask: How far can we go? In *Cryptographic Hardware and Embedded Systems - CHES 2013*, Springer.

[12] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.-S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., and Chee, S. HIGHT: A new block cipher suitable for low-resource device. In *Cryptographic Hardware and Embedded Systems - CHES 2006* (2006), L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 46–59.

[13] Karakoç, F., Demirci, H., and Harmanci, A. E. Biclique cryptanalysis of LBlock and TWINE. *Inf. Process. Lett. 113*, 12 (2013), 423–429.

[14] Rechberger, C. On bruteforce-like cryptanalysis: New meet-in-the-middle attacks in symmetric cryptanalysis. In *International Conference on Information Security and Cryptology (ICISC 2012)* (2012), T. Kwon, M.-K. Lee, and D. Kwon, Eds., vol. 7839 of *Lecture Notes in Computer Science*, Springer, pp. 33–36.

[15] Suzaki, T., Minematsu, K., Morioka, S., and Kobayashi, E. Twine: A lightweight, versatile block cipher. In *Proceedings of ECRYPT Workshop on Lightweight Cryptography (2011)*.

[16] Suzaki, T., Minematsu, K., Morioka, S., and Kobayashi, E. TWINE : A lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography (SAC 2012)* (2013), L. R. Knudsen and H. Wu, Eds., vol. 7707 of *Lecture Notes in Computer Science*, Springer, pp. 339–354.

[17] Tao, B., and Wu, H. Improving the biclique cryptanalysis of aes. In *Information Security and Privacy*, E. Foo and D. Stebila, Eds., vol. 9144 of *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 39–56.

[18] Wen, L., Wang, M., Bogdanov, A., and Chen, H. Note of Multidimensional MITM Attack on 25-Round TWINE-128 . Cryptology ePrint Archive, Report 2014/425, 2014.