A Higher Order Key Partitioning Attack with Application to LBlock

Riham AlTawy, Mohamed Tolba, and Amr M. Youssef

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

Abstract. In this paper, we present a higher order key partitioning meet-in-the-middle attack. Our attack is inspired by biclique cryptanalysis combined with higher order partitioning of the key. More precisely, we employ more than two equally sized disjoint sets of the key and drop the restrictions on the key partitioning process required for building the initial biclique structure. In other words, we start the recomputation phase of the attack from the input plaintext directly, which can be regarded as a Meet-in-the-Middle-attack where the tested keys have a predefined relation. Applying our approach on LBlock allows us to present a known plaintext attack on the full thirty two round cipher with time complexity of $2^{78.338}$ and negligible memory requirements. The data complexity of the attack is two plaintext-ciphertext pairs, which is the minimum theoretical data requirements attributed to the unicity distance of the cipher. Surprisingly, our results on the full LBlock are better, in terms of both computational and data complexity, than the results of its biclique cryptanalysis.

Keywords: Cryptanalysis, Meet-in-the-middle, Low data complexity, LBlock, Bicliques.

1 Introduction

Bicliques are structures that provide a formal representation of the initial execution separation in MitM attacks [14]. These structures have become particularly important after they have been used to present a key recovery attack on the full round Advanced Encryption Standard (AES) [5]. Indeed, a biclique attack is an optimized exhaustive search attack where the whole key space is tested efficiently. Accordingly, this class of attacks is usually used to analyze the full round cipher unlike various other attacks which can only be applied to reduced round versions. As a result of the exhaustive search nature of biclique cryptanalysis, attacks employing these structure are characterized by their high computational complexity which can reach that of the brute force search. However, practical gain has been shown in a dedicated FPGA implementation of the attack on AES [4]. Most of the biclique attacks require high data complexity (depending on the length of the employed biclique) which can sometimes reach the entire codebook.

The need for efficient lightweight cryptography is on the rise due to the current popularity of lightweight devices such as RFID chips and wireless sensor networks. Indeed, these systems provide convenient affordable services on tiny resource constrained environments. On the other hand, these systems must guarantee certain security and privacy requirements. More precisely, the adopted primitives must fulfill the aggressive restrictions of the application environment and at the same time maintain acceptable security margins. PRESENT [6], KATAN and KTANTAN [13], LED [16], Zorro [15], and LBlock [26] are some examples of cipher designs that have been proposed to address the needs of lightweight cryptography. Most of the recent cryptanalytic attacks on lightweight ciphers aim to analyze how some design concepts which are proposed for this environments have weakened these ciphers and broadened the effect of certain types of attacks. [20, 2, 19, 7, 23].

Recently, there has been an increased interest in adopting low data complexity attacks for the analysis of ciphers. This motivation is backed by the fact that security bounds are better perceived in a realistic model [8, 10]. More precisely, in a real life scenario, security protocols impose restrictions on the amount plaintext-ciphertext pairs that can be eavesdropped and/or the number of queries permitted under the same key. Given the fact that biclique cryptanalysis is characterized by its high data complexity, it has been implicitly avoided in the analysis of lightweight primitives.

In this work, we present a higher order key partitioning MitM attack. Our approach adopts only the recomputation phase from the biclique attack and does not require any specific initial biclique structure. Accordingly, we drop all the restrictions imposed by the bicliques on how the key is partitioned, and allow the use of related keys that would have been impossible otherwise. The absence of the biclique results in a low data complexity related key MitM attack in the single key setting in which the whole key space is searched efficiently through partial matching by recomputation [5]. To minimize the computational complexity of the recomputation, we employ a higher order number of disjoint sets of the master key [24]. More precisely, we partition the key space into more than two related keys (not necessarily independent related key differentials). Adopting this divide and conquer approach means that we have to deal with multiple small recomputed sets instead a dominating large set. We apply this attack on the lightweight block cipher LBlock which, similar to other lightweight ciphers, employs a simple key schedule with relatively slow diffusion to meet the resources constraints. Additionally, it adopts round subkeys that are shorter than the master key and a nibble-wise permutation. This fact allows our attack to achieve more gain over its biclique cryptanalysis counterpart [25]. Moreover, our attack on LBlock results in the minimum data requirements which makes it valid on RFID-like systems where the attacker can only acquire a very limited amount of plaintext-ciphertext pairs.

The rest of the paper is organized as follows. In the next section, we give a brief overview on the basic biclique attack. Afterwards, in Section 3, we give the specification of the lightweight block cipher LBlock. In Section 4, we provide the details of our approach and its application on LBlock. Specifically, we present a low-data complexity attack on the full thirty two round cipher. Finally, the paper is concluded in Section 5.

2 Biclique Cryptanalysis

Biclique cryptanalysis [5] was first used to present an accelerated exhaustive search on the full round AES. The basic idea of bicliques is to increase the number of rounds of the basic MitM attack by providing a formal representation of the initial structure and recomputing only the updated parts of the state. The key recovery attack starts by dividing the master key space into key sets where each key set K, is used to build one biclique. As depicted in Figure 1, a d-dimensional biclique is a structure of two sets of states P_i and S_j where $|P_i| = |S_j| = 2^d$ states and a key set K where $|K| = 2^{2d}$ keys which encrypt each state in P_i to each state in S_j. K is partitioned into three disjoint sets of key bits, i.e., $K = \{K_s, K_1, K_2\}$. Let $Enc_{[u,i,j]}(P^u_{i=0})$ and $Dec_{[u,i,j]}(S^u_{j=0})$ denote the encryption and decryption of the states $P^u_{i=0}$ and $S^u_{j=0}$ using the u, i, and j values of K_s, K_1, and K_2, respectively. These key sets are chosen such that for a given u of the $2^{|K_s|}$ values and all of i and j of the $2^{|K_1|}$, and $2^{|K_2|}$ values, respectively, $S^u_j = Enc_{[u,i,j]}(P^u_i)$, where $i, j \in \{0, .., 2^d - 1\}$.

The construction of bicliques imposes restrictions on the choices of K_1 and K_2 as they must result in independent related key differentials. In other words, K_1 and K_2 must be chosen such that the state variables between P^u_i and S^u_j that are affected by a change in the value of K_1 are different than those affected by a change in the value of K_2. In other words, a biclique can be constructed if for all u, i, and j of the $2^{|K_s|}, 2^{|K_1|}$, and $2^{|K_2|}$ values, respectively, the computation of $S^u_j = Enc_{[u,0,j]}(P^u_{i=0})$ does not share any active nonlinear state variables with the computation $P^u_i = Dec_{[u,i,0]}(S^u_{j=0})$.

The MitM key recovery attack using d-dimension bicliques starts with partitioning the master key space into $2^{|K|-2d}$ groups, where each group $K[u, i, j]$ has a single value u of the $2^{|K_s|}$ values and iterates over the 2^{2d} values of i and j of the $2^{|K_1|}$, and $2^{|K_2|}$ values, respectively. As depicted in Figure 1, usually the constructed biclique is placed on the plaintext side. The attack is divided into two main parts:

Biclique construction: at this stage, one searches for two independent related key differentials to partition the key into key groups consisting of a given K_s and all the values of K_1 and K_2. Since we do not use any biclique structures in our attack, we refer the reader to [5] for the detailed procedure for building the bicliques.

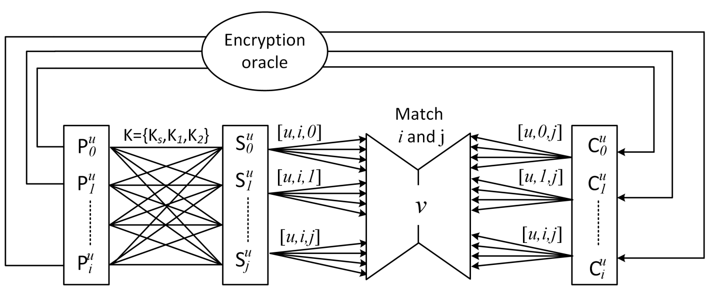

Fig. 1. Bicliques used in a MitM attack

Recomputation for MitM partial matching: during this step, partial state knowledge is computed from both the backward and forward directions to test each key group in an efficient manner. More precisely, in what follows, we give the steps performed for each key group $K[u, i, j]$.

- Choose an appropriate matching variable v between the end of the biclique and the ciphertext.
- *Forward recomputation*: for each j out of the 2^d values, do the following:
  - Compute the matching variable $\overrightarrow{v}^u_{0,j} = Enc_{[u,0,j]}(S^u_j)$ and store all the intermediate states.
  - For all the $2^d - 1$ values of i, compute the matching variable $\overrightarrow{v}^u_{i,j}$ by recomputing only those variables that differ from those previously computed using $K[u, 0, j]$ due to the effect of i.
- *Backward recomputation*: for all the 2^d values of P^u_i, ask the encryption oracle for their corresponding ciphertexts C^u_i.
- For each i of the 2^d values, do the following:
  - Compute the matching variable $\overleftarrow{v}^u_{i,0} = Dec_{[u,i,0]}(C^u_i)$ and store all the intermediate states.
  - For all $2^d - 1$ values of j, compute the matching variable $\overleftarrow{v}^u_{i,j}$ by recomputing only those variables that differ from those previously computed using $K[u, i, 0]$ due to the effect of j.

The remaining candidate keys $K[u, i, j]$ are those producing $\overrightarrow{v}^u_{i,j} = \overleftarrow{v}^u_{i,j}$. The surviving candidate keys should be further rechecked for full state matching as some of them could be false positives.

Testing each key group by the previous procedure has proved to lead to some improvements on the computational complexity. While all the 2^{2d} values of the key group are tested, we get three sets of computations. More precisely, the state variables that are affected by K_s only are computed once, those affected by either K_1 or K_2 are computed 2^d times, and the dominating large set is due to the variables that are influenced by both K_1 and K_2 which are recomputed 2^{2d} times. The data complexity of the attack is upper bounded by all possible values of different plaintext produced by all the bicliques $= min(2^{|K_s|+|K_1|}, 2^{\#\text{ of active bits in plaintext}})$.

The memory complexity is upper bounded by the memory required to store the forward and backward 2^d intermediate states $\approx 2^{d+1}$.

3 Description of LBlock

LBlock [26] is a 64-bit lightweight cipher with an 80-bit master key. It employs a 32-round Feistel structure and its internal state is composed of eight 4-bit nibbles. As depicted in Figure 2, the round function adopts three nibble oriented transformations: subkey mixing, 4-bit Sboxes, and nibble permutation. The 80-bit master key, K, is stored in a key register denoted by $k = k_{79}k_{78}k_{77}.......k_1k_0$. The leftmost 32 bits of the register k are used as i^{th} round subkey Sk_i. The key register is updated after the extraction of each Sk_i as follows:

1. $k \lll 29$.
2. $[k_{79}k_{78}k_{77}k_{76}] = S_9[k_{79}k_{78}k_{77}k_{76}]$.
3. $[k_{75}k_{74}k_{73}k_{72}] = S_8[k_{75}k_{74}k_{73}k_{72}]$.
4. $[k_{50}k_{49}k_{48}k_{47}k_{46}] \oplus [i]_2$,

where S_8 and S_9 are two 4-bit Sboxes. For further details, the reader is referred to [26].

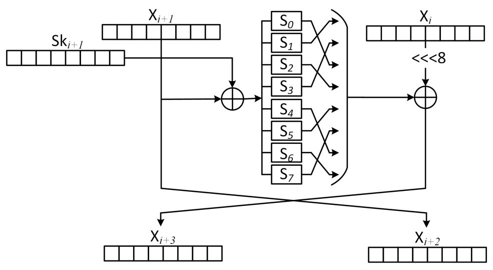

Fig. 2. The LBlock round function

LBlock [26] has been analyzed with respect to various types of attacks including impossible differential [17, 18, 9], integral [23, 22], MitM [1], boomerang [12], and biclique cryptanalysis [25]. Particularly, the attack presented in [25] is a typical high data complexity biclique cryptanalysis where the authors presented an attack with a time complexity $= 2^{78.4}$ and a data complexity of 2^{52}. Our result for the 9th order key partitioning MitM cryptanalysis of the full round LBlock has a better time complexity and is launched with only two known plaintext-ciphertext pairs. In Table 1, we provide a summary of the current cryptanalytic results on the LBlock cipher in the single key model. In what follows, we give the notation used in our attack.

Attack	#Rounds	Time	Memory	Data	Reference
Integral	22	2^{70}	2^{63}	2^{61} CP	[22]
Impossible DC	22	$2^{79.28}$	-	2^{58} CP	[17]
	23	$2^{75.36}$	2^{74}	2^{59}CP	[9]
Biclique cryptanalysis	32	$2^{78.4}$	2^{8}	2^{52} CP	[25]
Meet-in-the-middle	32	$2^{78.338}$	2^{7} FC	**2 KP**	This paper

Table 1. Summary of the current cryptanalytic results on LBlock. DC, CP, KP, and FC stands for Differential Cryptanalysis, Chosen Plaintext, Known Plaintext, and Full Computation, respectively.

3.1 Notation

The following notation will be used throughout the remainder of the paper.

- K: The master key.
- Sk_i: i^{th} round sub key.
- X_i: The eight 4-bit nibble state at round i.
- $X_i[j]$: j^{th} nibble of the i^{th} round state.
- $K_{[i,j]}$: i^{th} and j^{th} bits of master key K.

4 Higher Order Key Partitioning MitM Attack

While most of the previous works [3, 11, 24] were trying to decrease the length and/or dimension of the bicliques in order to reduce the data complexity, we opted for removing the biclique structure completely from our attack. In the sequel, we turn the biclique attack into a MitM attack where the whole key space is efficiently tested. However, in contrast to the basic MitM attack, the same tested key is used to compute the matching variable from both the forward and backward directions. Similar to MitM attacks, our attack is a known plaintext attack where the number of required plaintext-ciphertext pairs is solely determined by the relationship between the block length and key lengths. Hence, given its negligible memory complexity, this approach provides an actual computational gain over exhaustive search as both of them have the same data requirements.

Our attack skips the independence requirements imposed on the choice of the related keys in the biclique attack and starts the recomputation phase from the plaintext. Thus, our approach is equivalent to a MitM attack where the tested keys have a predefined relation. Moreover, we consider a divide and conquer approach where higher order partitioning of the key space is adopted. In other words, instead of dividing the key K into three disjoint sets, we divide it into $n+1$ sets with $n > 2$ to minimize the complexity of the recomputation in both directions. Given that the key is partitioned into a $(|K| - nd)$-bit set and n d-bit sets, adopting this higher order partitioning shares the complexity of the attack between $n+1$ sets of recomputations where Sboxes of the i^{th} set are recomputed $2^{(i-1)d}$ times, $i \in \{1, 2, ..., n+1\}$.

In the sequel, we apply this technique on LBlock and present a low data complexity key recovery attack on the full round cipher. In fact, our best obtained result is a two known plaintext MitM attack where the key is partitioned into nine 4-bit sets (i.e., n=9). However, based on our trials with different values of n, we expect that further reduction in the computational complexity can be obtained with higher values of n, but given our available computational resources, the complexity of finding the optimal attack parameters as n grows is not practical. Our results are particularly interesting, because they show that removing the biclique structure from biclique-like attacks can have a good impact on both the data and computational complexities in some cases as with the case of LBlock.

4.1 A Low Data Complexity Attack on LBlock

In this section, we present a low data complexity attack on the full round LBlock. The attack exploits the weak diffusion of the key schedule. This fact enables us to partition the master key into higher order related key partitions for our MitM attack on the full cipher with some gain over the biclique attack [25]. With the aim of minimizing the computational complexity, we used an exhaustive search algorithm to test all possible 4-bit partitioning possibilities and different matching variables. Our search algorithm shows that partitioning K into K_s, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8,and K_9 (i.e., $n = 9$) results in a MitM attack with a time complexity of $2^{78.338}$ and a memory complexity of $\approx 2^7$. The parameters of our best case for the MitM key recovery attack are as follows:

– Matching round: 22.
– Matching values: $X_{22}[2]$ and $X_{22}[7]$.
– $K_1 = K_{[76,75,74,73]}$
– $K_2 = K_{[59,58,57,56]}$
– $K_3 = K_{[38,37,36,35]}$.
– $K_4 = K_{[30,29,28,27]}$.
– $K_5 = K_{[25,24,23,22]}$.
– $K_6 = K_{[17,16,15,14]}$.
– $K_7 = K_{[13,12,11,10]}$.
– $K_8 = K_{[9,8,7,6]}$.
– $K_9 = K_{[5,4,3,2]}$.

MitM attack with n=3. Due to the complexity of visualizing our best obtained case (i.e., $n = 9$), in what follows, we demonstrate the details of the attack in the simplest case when K is partitioned into three related partitions (n=3). The algorithm indicates that the best partitioning of the master key for the case of $n = 3$ is when $K_1 = K_{[25,24,23,22]}$, $K_2 = K_{[13,12,11,10]}$, $K_3 = K_{[3,2,1,0]}$, and $K_s = K - \{K_1, K_2, K_3\}$.

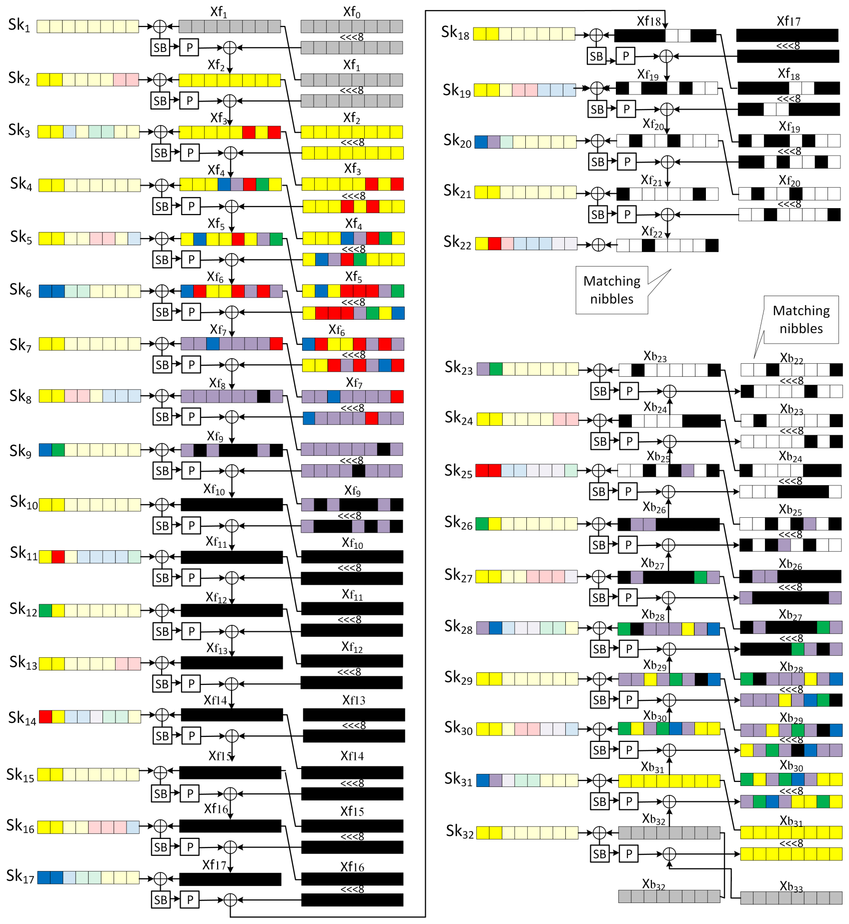

Fig. 3. Third order partitioned MitM attack on the thirty two rounds LBlock

Accordingly, in our attack we test 2^{68} key groups where each group has 2^{12} keys, all of which have one value of K_S and differ in the values of K_1, K_2, and K_3. Figure 3 depicts the recomputation process adopted for our MitM attack for each key group. Our search algorithm shows that choosing the matching variable v as the two nibbles $X_{22}[2,7]$ results in the best computational complexity for our attack. In the sequel, given one known plaintext-ciphertext pair, we evaluate the matching variable v from both the forward and backward directions with the same key and discard keys that produce $\overrightarrow{v} \neq \overleftarrow{v}$.

The computational complexity of the LBlock round function is dominated by the Sbox lookups. Consequently, to determine the gain of our approach, we use the number of Sboxes that are calculated in both directions relative to the 318

Sbox lookups used in the full thirty two rounds computation. We distinguish the color scheme used in Figure 3 as follows:

- Gray: input nibbles which are either plaintext or ciphertext and these nibbles remain constant while testing the whole keyspace.
- Yellow: are the nibbles that are affected by changing the value of K_s only. Accordingly, when testing the 2^{12} keys within each key group, these nibbles are evaluated once.
- Red, blue, and green: are those nibbles that are influenced by a change in either K_1, K_2, or K_3, respectively. In other words, for every tested value of K_1, the values of the red nibbles are updated while the values of the blue and green nibbles remain unchanged. Same rationale applies for K_2 and K_3. Consequently, these nibbles are computed 2^4 times for each key group.
- Purple: nibbles that are affected by changing any two keys. For example, if K_2 and K_3 changed, then the values of all the blue and green nibbles, and the values of the purple nibbles that depend on both keys should be updated. Purple nibbles are recomputed 2^8 times within one key group.
- Black: are the nibbles that are affected by a change in the values of all the three keys and these are evaluated 2^{12} times.
- White: are those nibbles whose values do not affect the value of the matching variable and thus we do not need to compute them in our attack.

Since our gain is estimated based on the number of Sbox lookups, the six rightmost nibbles in the round subkeys are faded in color because only the two leftmost nibbles are the ones that count in our calculations. In what follows, we give the details of the forward and backward recomputation used for testing the 2^{12} keys within a given key group. However, since each key group has one value of K_s and 2^4 values for each K_1, K_2, and K_3, we denote the tested key by $K[i, j, l]$ where each i, j, and l is one of the 2^4 values of K_1, K_2, and K_3, respectively. We also denote the matching variable that is computed using $K[i, j, l]$ as $v_{[i,j,l]}$.

Forward recomputation: As depicted in Figure 3, the forward computation spans over states Xf_2 to Xf_{22}. The 64-bit input plaintext P is loaded in states Xf_0 and Xf_1. We now need to partially encrypt the plaintext P with all the 2^{12} keys $K[i, j, l]$ to derive 2^{12} values for the matching variable $\overrightarrow{v_{i,j,l}}$ which is the 8-bit output at $Xf_{22}[2, 7]$. We first evaluate the matching variable $\overrightarrow{v_{[0,0,0]}} = Enc_{K[0,0,0]}(P)$ and store the computations. Now we do the same for all the $2^4 - 1$ values of each key partition at a time. More precisely, we compute $\overrightarrow{v_{[i,0,0]}} = Enc_{K[i,0,0]}(P)$, $\overrightarrow{v_{[0,j,0]}} = Enc_{K[0,j,0]}(P)$, and $\overrightarrow{v_{[0,0,l]}} = Enc_{K[0,0,l]}(P)$ for all i, j, and $l \in \{1, .., 2^4 - 1\}$, and store these computations as well. However, during these three computations, we only evaluate the nibbles that are different from the first stored computation using $K[0, 0, 0]$ due to the effect of either i, j, or l. These are the red, blue, green, purple, and black nibbles. Moreover, to test any two key partitions combination, e.g., $K[i, j, 0]$, we only recompute the values of the purple and black nibbles that differ from that of the stored computations using $K[i, 0, 0]$ and $K[0, j, 0]$ (i.e., where the effect of i and j overlap). Lastly, when testing any three key partitions combination, we only recompute those nibbles where the nibbles at their corresponding positions in the computations of $\overrightarrow{v_{[i,0,0]}}$, $\overrightarrow{v_{[0,j,0]}}$, and $\overrightarrow{v_{[0,0,l]}}$ overlap. As shown in Figure 3, the forward recomputation for one key group requires computing 54 Sboxes once, 25 Sboxes 2^4 times, 21 Sboxes 2^8 times, and 86 Sboxes 2^{12} times.

Backward recomputation: The backward computation is depicted on the right side of Figure 3, where states Xb_{31} to Xb_{22} are iteratively recomputed to generate the matching variable. We use the ciphertext C corresponding to the plaintext P in states Xb_{32} and Xb_{33}. In the sequel, we proceed with partially decrypting C using the 2^{12} keys to evaluate the 2^{12} matching variable values $\overleftarrow{v_{i,j,l}}$ following the same procedure used in the forward recomputation. As depicted in Figure 3, the backward recomputation for one key group requires computing 24 Sboxes once, 14 Sboxes 2^4 times, 18 Sboxes 2^8 times, and 24 Sboxes 2^{12} times.

Surviving candidates: As we are testing 2^{12} keys and the matching size is 8-bits, then for each key group we get 2^4 potential candidates, which need to be further retested for full state matching. This process of retesting the surviving candidates requires 2^4 full LBlock computations. Moreover, even after testing the whole key space, the relation between the key size k and the state size b determines the number of the remaining potentially right keys. If $b = k$, then only the right key remains and no further testing is required. However, in LBlock, the master key length is larger than the state size and hence, we end up with $2^{k-b} = 2^{16}$ potentially right keys. In this case, to recover the right key, the data complexity of the MitM attack is $\lceil \frac{k}{b} \rceil = 2$ plaintext-ciphertext pairs. In other words, these 2^{16} keys must be further retested with an additional plaintext-ciphertext pair.

Complexity: The computational complexity of the attack is evaluated based on the number Sbox lookups that are required in the forward and backward recomputations and testing surviving keys within a key group. The whole attack tests 2^{68} key groups, each requires the computations of $54+2^4(25)+2^8(21)+2^{12}(86) = 358086$ Sboxes in the forward direction, $24+2^4(14)+2^8(18)+2^{12}(24) = 103160$ Sboxes in the backward direction, and $2^4(318)$ Sboxes for retesting candidate keys. Accordingly, the overall computational complexity of the attack is given by $2^{68}((358086 + 103160)/318 + 2^4) \approx 2^{78.518}$. The memory complexity is upper bounded by storing 3×2^4 full LBlock computations, which is practically negligible, and the data complexity is two known plaintext-ciphertext pairs.

Generally, when adopting n key partitions of dimension d bits and m-bit matching variable, the same forward and backward recomputation procedures is applied. However, in this case, we get $n + 1$ recomputation sets. More formally, let Sb_i denote the number of Sboxes that belong to the i^{th} recomputation set, $1 \leq i \leq n + 1$, the computational complexity of the attack is given by:

$$2^{|K|-nd}\left(\frac{\sum_{i=1}^{n+1}(2^{(i-1)d} \times Sb_i)}{318} + 2^{nd-m}\right),$$

and the memory complexity is given by $n \times 2^d$. Accordingly, given the parameters of our best obtained result when $n = 9$, the time and memory complexity of the MitM attack on the full LBlock is $2^{78.338}$ and $\approx 2^7$ full computations, respectively.

5 Conclusion

In this work, we have presented a higher order key partitioning MitM attack. Our technique adopts the recomputation phase from the biclique attack without using its initial structure. We have shown that in some cases removing the biclique structure from the MitM attack can lead to better computational and data complexity as with the case of LBlock. This fact is attributed to the restrictions imposed by the biclique on how the master key can be partitioned. On the other hand, if we search for the best key partitioning for a minimum complexity MitM attack and begin the recomputation phase from the first round, we can get more savings. Moreover, we adopted a divide and conquer approach where the key space is divided into more than two related key sets. Thus, the computational complexity of the forward and backward recomputations is shared among multiple smaller sets and not being dominated by a large one. We applied our approach on LBlock and presented a low data complexity MitM attack on the full round cipher. Our best obtained result on the full round LBlock is a known plaintext MitM attack where the key is partitioned into nine related key sets. This attack has a time, memory, and data complexity of $2^{78.338}$, 2^7 full computations, and 2 known plaintexts, respectively, which are better than the results obtained by the biclique cryptanalysis of the cipher [25].

Inspired by biclique cryptanalysis, our attack can be described as a bruteforce-like cryptanalysis [21] which is not able to conclude that a particular primitive has some cryptanalytic weakness, as in general it covers the whole cipher. However it can help to better understand the real security provided by the primitive when no attack tweaks are adopted. Most of the applications of bruteforce-like cryptanalysis have an advantage that is sometimes much smaller than a factor of 2. Nevertheless, for lightweight ciphers with key sizes of 80 bits or less, this is very useful to know, especially when the gain compared to the optimized bruteforce search is even a factor 2. To this end, designers of lightweight symmetric primitives should be motivated to consider this class of attacks during the assessment of any new proposed design.

6 Acknowledgment

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 by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. AlTawy, R., and Youssef, A. M. Differential sieving for 2-step matching meet-in-the-middle attack with application to LBlock. In *Lightsec* (2014), T. Eisenbarth and E. Öztürk, Eds., vol. 8898 of *Lecture Notes in Computer Science*, Springer, pp. 126–139.
2. Bar-On, A., Dinur, I., Dunkelman, O., Lallemand, V., and Tsaban, B. Improved analysis of zorro-like ciphers. Cryptology ePrint Archive, Report 2014/228, 2014. http://eprint.iacr.org/.
3. Bogdanov, A., Chang, D., Ghosh, M., and Sanadhya, S. K. Bicliques with minimal data and time complexity for AES (extended version). Cryptology ePrint Archive, Report 2014/932, 2014. http://eprint.iacr.org/.
4. Bogdanov, A., Kavun, E., Paar, C., Rechberger, C., and Yalcin, T. Better than brute-force–optimized hardware architecture for efficient biclique attacks on AES-128. In *ECRYPT Workshop, SHARCS-Special Purpose Hardware for Attacking Cryptographic Systems* (2012).
5. Bogdanov, A., Khovratovich, D., and Rechberger, C. Biclique cryptanalysis of the full AES. In *ASIACRYPT* (2011), D. Lee and X. Wang, Eds., vol. 7073 of *Lecture Notes in Computer Science*, Springer, pp. 344–371.
6. 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 *CHES* (2007), P. Paillier and I. Verbauwhede, Eds., vol. 4727 of *Lecture Notes in Computer Science*, Springer, pp. 450–466.
7. Bogdanov, A., and Rechberger, C. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In *SAC* (2011), A. Biryukov, G. Gong, and D. Stinson, Eds., vol. 6544 of *Lecture Notes in Computer Science*, Springer, pp. 229–240.
8. Bouillaguet, C., Derbez, P., Dunkelman, O., Fouque, P.-A., Keller, N., and Rijmen, V. Low-data complexity attacks on AES. *IEEE Transactions on Information Theory 58*, 11 (2012), 7002–7017.
9. Boura, C., Minier, M., Naya-Plasencia, M., and Suder, V. Improved impossible differential attacks against round-reduced LBlock. Cryptology ePrint Archive, Report 2014/279, 2014. http://eprint.iacr.org/.
10. Canteaut, A., Naya-Plasencia, M., and Vayssire, B. Sieve-in-the-middle: Improved MITM attacks. In *CRYPTO* (2013), R. Canetti and J. Garay, Eds., vol. 8042 of *Lecture Notes in Computer Science*, Springer, pp. 222–240.
11. Chang, D., Ghosh, M., and Sanadhya, S. Biclique cryptanalysis of full round AES with reduced data complexity, 2013.
12. Chen, J., and Miyaji, A. Differential cryptanalysis and boomerang cryptanalysis of LBlock. In *Security Engineering and Intelligence Informatics* (2013), A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, and L. Xu, Eds., vol. 8128 of *Lecture Notes in Computer Science*, Springer, pp. 1–15.
13. De Cannière, C., Dunkelman, O., and Knežević, M. KATAN and KTANTANa family of small and efficient hardware-oriented block ciphers. In *CHES* (2009), C. Clavier and K. Gaj, Eds., vol. 5747 of *Lecture Notes in Computer Science*, Springer, pp. 272–288.
14. Diffie, W., and Hellman, M. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer 10*, 6 (1977), 74–84.
15. Gérard, B., Grosso, V., Naya-Plasencia, M., and Standaert, F.-X. Block ciphers that are easier to mask: how far can we go? In *CHES* (2013), G. Bertoni

and J.-S. Coron, Eds., vol. 8086 of *Lecture Notes in Computer Science*, Springer, pp. 383–399.

16. Guo, J., Peyrin, T., Poschmann, A., and Robshaw, M. The LED block cipher. In *CHES* (2011), B. Preneel and T. Takagi, Eds., vol. 6917 of *Lecture Notes in Computer Science*, Springer, pp. 326–341.
17. Karakoç, F., Demirci, H., and Harmanc, A. Impossible differential cryptanalysis of reduced-round LBlock. In *Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems* (2012), I. Askoxylakis, H. Phls, and J. Posegga, Eds., vol. 7322 of *Lecture Notes in Computer Science*, Springer, pp. 179–188.
18. Liu, Y., Gu, D., Liu, Z., and Li, W. Impossible differential attacks on reduced-round LBlock. In *Information Security Practice and Experience* (2012), M. Ryan, B. Smyth, and G. Wang, Eds., vol. 7232 of *Lecture Notes in Computer Science*, Springer, pp. 97–108.
19. Mendel, F., Rijmen, V., Toz, D., and Varici, K. Differential analysis of the LED block cipher. In *ASIACRYPT* (2012), X. Wang and K. Sako, Eds., vol. 7658 of *Lecture Notes in Computer Science*, Springer, pp. 190–207.
20. Nakahara Jr, J., Sepehrdad, P., Zhang, B., and Wang, M. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In *Cryptology and Network Security* (2009), J. A. Garay, A. Miyaji, and A. Otsuka, Eds., vol. 5888 of *Lecture Notes in Computer Science*, Springer, pp. 58–75.
21. Rechberger, C. On bruteforce-like cryptanalysis: New meet-in-the-middle attacks in symmetric cryptanalysis. In *ICISC* (2012), T. Kwon, M.-K. Lee, and D. Kwon, Eds., vol. 7839 of *Lecture Notes in Computer Science*, Springer, pp. 33–36.
22. Sasaki, Y., and Wang, L. Comprehensive study of integral analysis on 22-round lblock. In *ICISC* (2013), T. Kwon, M.-K. Lee, and D. Kwon, Eds., vol. 7839 of *Lecture Notes in Computer Science*, Springer, pp. 156–169.
23. Sasaki, Y., and Wang, L. Meet-in-the-middle technique for integral attacks against Feistel ciphers. In *SAC* (2013), L. R. Knudsen and H. Wu, Eds., vol. 7707 of *Lecture Notes in Computer Science*, Springer, pp. 234–251.
24. Siavash Ahmadi, Zahra Ahmadian, J. M., and Aref, M. R. Low data complexity biclique cryptanalysis of block ciphers with application to Piccolo and HIGHT. Cryptology ePrint Archive, Report 2013/511, 2013. http://eprint.iacr.org/.
25. Wang, Y., Wu, W., Yu, X., and Zhang, L. Security on LBlock against biclique cryptanalysis. In *Information Security Applications* (2012), D. Lee and M. Yung, Eds., vol. 7690 of *Lecture Notes in Computer Science*, Springer, pp. 1–14.
26. Wu, W., and Zhang, L. LBlock: a lightweight block cipher. In *Applied Cryptography and Network Security* (2011), J. Lopez and G. Tsudik, Eds., vol. 6715 of *Lecture Notes in Computer Science*, Springer, pp. 327–344.