IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 9, 2019, accepted December 17, 2019, date of publication December 24, 2019, date of current version January 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962101

Generalized Meet in the Middle Cryptanalysis of Block Ciphers With an Automated Search Algorithm

SIAVASH AHMADI AND MOHAMMAD REZA AREF
Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
Corresponding author: Siavash Ahmadi (s_ahmadi@ee.sharif.edu)

This work was supported in part by the Iranian National Science Foundation (INSF) under Contract 96.53979, and in part by the INSF Cryptography Chair.

ABSTRACT Meet in the middle (MITM) attack is one of the most important and applicable methods for cryptanalysis of block ciphers. In this paper, a more generalized method for MITM attack is considered. For this purpose, a notion, namely cut-set, is utilized by which several numbers of MITM attacks can be performed. However, manual investigation on these cases is time-consuming and sometimes not error-free. Therefore, a new search algorithm is also provided to obtain proper attacks in a timely manner. For examination, this new search algorithm, which could make an automated attack along with some certain ideas, is applied on HIGHT, Piccolo-128, CRAFT and AES-128 block ciphers. The least time complexities which are obtained by generalized MITM attack on full HIGHT, Piccolo-128, CRAFT and AES-128 are $2^{125.08}$, $2^{126.78}$, $2^{123.25}$ and $2^{125.53}$, respectively. The results on full-round CRAFT are, to the best of our knowledge, the first cryptanalysis results in the single-key model except the designers' investigations. In addition, the results show some improvements for complexities of all the attacks, especially on HIGHT.

INDEX TERMS Automated attack, cryptanalysis, cut-set, meet in the middle.

I. INTRODUCTION

Block ciphers are usually the main primitive used for securing communications in various technologies such as cellular networks, internet of things, and so on. Therefore, their security evaluation remains as an important concenteration point of the cryptanalysts. One of the generic methods which is always applied for security evaluation of block ciphers is MITM attack. The basic method was first introduced in [1] for cryptanalysis of DES block cipher, and then many variants and improvement techniques for MITM attack were proposed. The basic MITM attack begins from a pair of plaintext/ciphetext, say starting states, and is completed by a matching technique at the middle of a target block cipher, say ending internal/intermediate state. However, some of enhanced versions of MITM attack have focused on improving the basic attack by choosing different starting states or promotion of attack structure such as 3-subset MITM [2], splice-and-cut [3], and biclique [4], [5] attacks. Some ideas also have concentrated on improvements of the matching technique at the ending internal/intermediate state of MITM attack such as early abort [6], sieve in the middle [7] or indirect partial matching [3] techniques.

Recently, automation of cryptanalysis methods has become a new important approach for two below reasons:

1) From the designers' point of view, it quickly gives the ability to ensure the security of a newly designed block cipher against various automated attacks;
2) For attackers, it rapidly gives a proper way for analyzing a target block cipher against such attacks.

The MITM attack is also not an exception. As a literature review, there are various works on proposing automation procedures for different types of attacks on block ciphers, such as linear [8], [9], differential [8]–[11], impossible differential [12], [13], MITM [13], [14], biclique [15], and division property-based [16] attacks. The automation usually performs by an exhibition of a framework in which a block cipher should be defined and then it will automatically analyze the cipher and return the results. Each framework has been mostly designed for a specific attack and it does not often perform

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Verticale.

TABLE 1. Results comparison of generalized MITM attack on various block ciphers with the best previous ones.

Block Cipher	Time	Data	Memory	Reference
HIGHT(Full)	$2^{125.67}$	2^{42}	2^{16}	[22]
HIGHT(Full)	$2^{125.2}$	2^{24}	2^{16}	Sec. V-A2
Piccolo-128(Full)	$2^{127.12}$	2^{4}	2^{8}	[17]
Piccolo-128(Full)	2^{127}	2^{40}	2^{12}	Sec. V-B1
CRAFT(Full)	2^{124}	2^{4}	2^{4}	[20]
CRAFT(Full)	$2^{126.53}$	2^{48}	2^{28}	Sec. V-C1
CRAFT(Full)	$2^{123.25}$	2^{56}	2^{12}	Sec. V-C1
AES-128(Full)	$2^{125.56}$	2^{128}	2^{8}	[23]
AES-128(Full)	$2^{126.01}$	2^{72}	2^{16}	[24]
AES-128(Full)	$2^{125.56}$	2^{128}	2^{32}	Sec. V-D1

the whole attack procedure, but it usually tries to compute the attack complexities by some different or innovative methods.

A. CONTRIBUTIONS

The purpose of this paper is to express a generalized MITM attack, which can be considered as a general form of some previous attacks, in a way that it could be implemented by an automated search method. This is proposed by enjoying the notion of cut-set which was defined in [17]. Firstly, it should be cleared that how the notion of cut-set can be utilized in an attack and what its benefits are. Secondly, it should be illustrated that how the procedure of computing complexities for the attack can be automatically performed. Finally, some targetted block ciphers should be checked by the automated generalized MITM to show its performance. Although the generalized MITM attack is not limitted to any class of block ciphers, in this paper HIGHT [18], Piccolo-128 [19], CRAFT [20] and AES-128 [21] block ciphers are chosen as its targets. The results and comparisons are shown in Table 1. As it can be seen in Table 1, the lowest time complexity is for generalized MITM attack in all cases. Also, it should be noted that all complexities obtained for HIGHT block cipher are less than or equal to the previous best result. Recently, some cryptanalysis results for CRAFT are presented [25]–[27], however, they are not on the full-round version of CRAFT and single-key model as ours.

B. RELATED WORKS

There are many related works to MITM attack on block ciphers such as [2]–[4], [13]–[15], [17], but, only some of them have automation approach just like generalized MITM attack. One of them is Bouillaguet's work [28], which has concentrated on proposing efficient basic MITM attacks on (mostly reduced-round) AES-based primitives by proposing an automatic search for recursive combinations of solvers. However, the most important one is Derbez's work [13] which helped matching part of basic MITM attack to be quickly implemented. The main idea of [13] for MITM attack is to find an equation for matching part with the lowest independent variables from the both plaintext-side and ciphertext-side. With these equations from each side, the extra computations can be omitted and the required computations can be extracted. This only works for the matching part of a basic MITM attack.

Most of the other works have focused on proposing new attacks rather than introducing search algorithms to obtain the most efficient attack.

C. OUTLINE

The rest of the paper is organized as follows: generalized MITM attack is introduced in Section II. Then, it is shown in Section III that the generalized MITM attack can cover some traditional versions of MITM attacks. In Section IV, the exact automation procedure for generalized MITM attack is explained. Afterwards, some experimental results are also provided in Section V. Finally, the conclusion is given in Section VI.

II. GENERALIZED MITM ATTACK

A. BASIC ATTACK

The basic MITM attack starts from one known plaintext/ciphertext pair (P, C) and a chosen matching variable V often with the smallest size of operations in the cipher (e.g. a bit, nibble, byte or etc., say word). Then according to the partial matching technique, the matching variable V has to be computed by effective subkeys from P and C, in forward and backward directions, respectively. Therefore, suppose that the master key bits have been partitioned into three parts of common (K^c), forward (K^f) and backward (K^b) bits in such a way that partial encryption from P to V only requires (K^c, K^f) bits, and partial decryption from C to V only requires (K^c, K^b) bits. Assuming a fixed value for K^c, forward and backward computations for calculation of the matching variable V are independent from each other. So, they should be computed and stored in a sorted format, separately. Finaly, any guessed value for master key bits with common value of matching variable from both side can be considered as a condidate for the master key.

Let C_f and C_b be the time complexity of the partial encryption and decryption, respectively. Then the upper bounds of time and memory complexity of the basic MITM attack are equal to $2^{|K^c|}(2^{|K^f|}C_f + 2^{|K^b|}C_b)$ and $2^{|K^f|} + 2^{|K^b|}$, respectively. In order to calculate more accurate time complexity, summation of each S-box complexity could be considered. The involved computational complexity for each S-box in the attack is equal to the number of guessed keys before computing that S-box. Hence, considering the complexity of non-linear parts as the dominant complexity, time complexity of the attack should be approximately equal to summation of each S-box complexity which should be normalized to the total number of S-boxes in the encryption algorithm.

B. PROPOSING GENERALIZED ATTACK

For a MITM attack, it is not mandatory to start from a plaintext or ciphertext to find the value of a matching variable in both directions. More precisely, any internal states can also be used as the starting state of the attack, while it may result

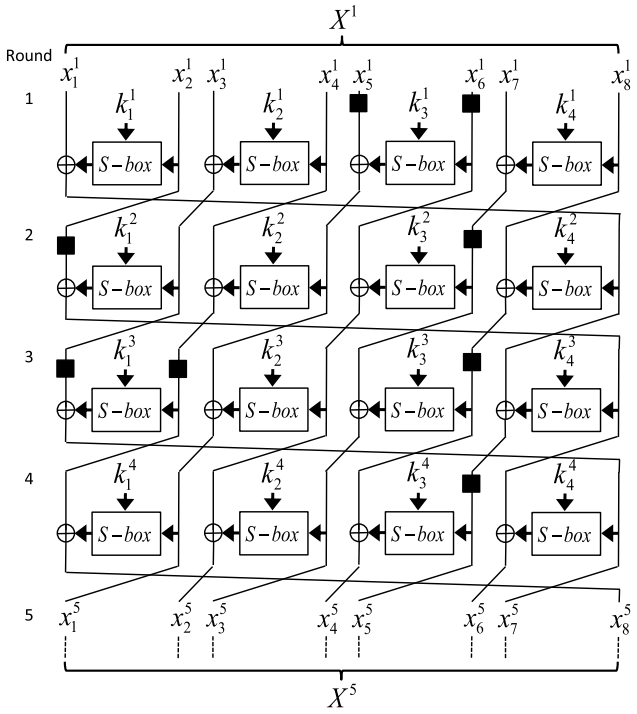


FIGURE 1. An example of cut-set in a 4-branch generalized Feistel structure.

in a more data complexity due to various plaintext/ciphertext pairs involved in the attack. In [17], the notion of cut-set has been defined which can be considered as a new starting state for MITM attack. Definition of cut-set is reminded as follows:

Definition 1 (Cut-set): Let b be the block size of a block cipher. A b-bit part of the intermediate states of the cipher from which the plaintext (or the ciphertext) can be calculated completely, provided that the cipher key is known, is called a cut-set.

An example of cut-set for a 4-branch generalized Feistel structure with nibble-wise computations is shown in Figure 1. Assuming that all the sub-keys k_j^i (in which i is round number and $1 \leq j \leq 4$) are known and the internal state of round i input is shown by $X^i = (x_1^i, x_2^i, \ldots, x_8^i)$ (in which $x_j^i, 1 \leq j \leq 8$ has 4-bit length), the set of $\mathcal{C} = (x_5^1, x_6^1, x_1^2, x_6^2, x_1^3, x_2^3, x_6^3, x_6^4)$ can form a cut-set. One can easily check that all the intermediate states of x_j^i can be computed by $\mathcal{C}$ members.

Now, according to the idea of using cut-sets, one can choose a random cut-set as the starting state of the basic MITM attack. Naturally, any pair of plaintext/ciphertext can also be considered as an starting/ending cut-set of a block cipher. The procedure of choosing the matching variable is the same as basic MITM attack. Hence, it can be seen that in this new setting, namely generalized MITM attack, the total number of possible attacks increases approximately with the number of possible cut-sets multiples by the number of possible matching variables.

Considering a cut-set as a starting state relaxes some limitations of basic MITM attack and let an adversary take the diffusion weaknesses of the block cipher and key schedule into accout. In the other words, the chosen cut-set can be found in such a way that the associated subkeys in forward and backward directions (between the cut-set and the matching variable) result in lower time complexity.

Generalized MITM attack can be expressed in four main steps as follows:

Step 1 (Initial Selection):

1-1. Choose a random cut-set $\mathcal{C}$ with an arbitrary fixed value by the intermediate states of the cipher.

1-2. Choose a proper matching variable $\mathcal{V}$, clearly not in the same place with any part of the chosen cut-set $\mathcal{C}$.

Step 2 (Pre-Processing):

2-1. Remove all the extra computations from both sides of the matching variable $\mathcal{V}$ which are not required for computing $\mathcal{V}$ from the cut-set $\mathcal{C}$ in forward and backward directions.

Step 3 ($\mathcal{C}$ to $\mathcal{V}$ Propagation (Perform for all Guessed Keys)):

3-1. For a chosen value of master key, perform partial encryption in forward direction from the cut-set $\mathcal{C}$ until computing the matching variable $\mathcal{V}$ as the pre-computations. If the computation of ciphertext is required, then the proper query should be sent to obtain the corresponding plaintext and then continue until the matching variable is computed. For the other guessed master keys, the recomputation technique must be used for each required bit of master key, step by step.

3-2. Perform partial decryption in backward direction from the cut-set $\mathcal{C}$ until computing the matching variable $\mathcal{V}$ similar to the forward direction.

Step 4 (Checking the guessed keys):

4-1. The matching variable values from both sides should be the same. Therefore, any guessed master key for which the matching variable values are the same for both side computations can be considered as a candidate for the correct master key. Else, the guessed key must be omitted.

It should be noted that it is possible to include the key schedule in the attack. This is usually necessary if the key schedule is not as simple as a permutation. In this scenario, first a random cut-set $\mathcal{K}$ with fixed value must be chosen from key schedule intermediate states as the master key K. Next, it should be propagated in both forward and backward directions of key schedule algorithm to compute all the states of key schedule or equivalently subkeys of the cipher (so in the propagation procedure, it will be obetained that which parts of the cut-set $\mathcal{K}$ have influenced on each subkeys; this will be used in the recomputation technique to calculate the computational complexities). Then the above procedure for generalized MITM attack can be performed.

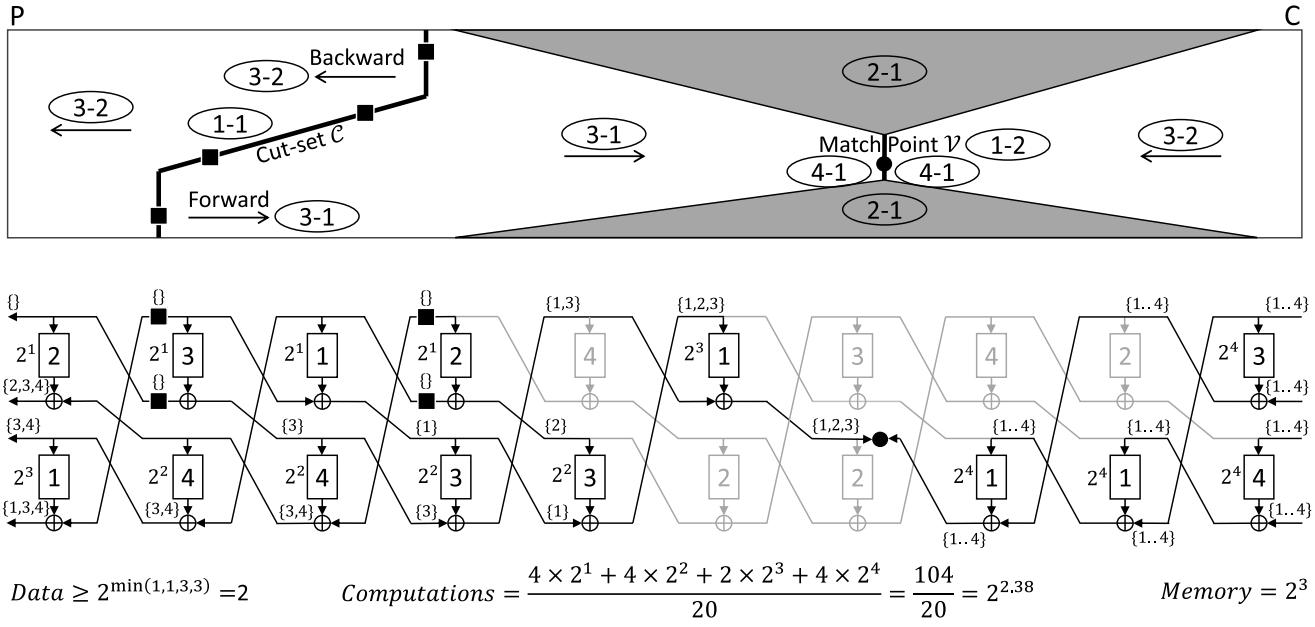


FIGURE 2. Generalized MITM attack on a toy cipher with 4-nibble block and 4-bit key sizes.

The time, memory and data complexities of generalized MITM attack can be obtained as follows:

- Time complexity: it is approximately equal to the summation of each S-box computation multiple by number of guessed master key involved in that computation, normalized to the total number of S-boxes in the cipher.
- Memory complexity: it is equal to the summation of each should-be-stored computation multiple by number of guessed directly involved bits on that computation.
- Data complexity: it is equal to the number of all the different queries during the attack.

The memory complexity of the attack can be improved for each specific attack according to the conditions and the cipher and key schedule algorithms. Moreover, $min(2^{min_i\{\#\mathcal{K}^i \to P\}}, 2^{min_i\{\#\mathcal{K}^i \to C\}})$ in which $\#\mathcal{K}^i \to P$ (resp. $\#\mathcal{K}^i \to C$) is the number of activated plaintext (resp. ciphertext) bits by i^{th} bit of cut-set $\mathcal{K}$, can be considered as the lower bound for data complexity of the generalized MITM attack. As an example, the total procedure of generalized MITM attack on a toy cipher with simple fixed subkeys is depicted in Figure 2. In this figure, cut-set $\mathcal{C}$ is shown by complete black squares and matching variable $\mathcal{V}$ is shown by a complete black circle. Each rectangle shows an 4×4 S-box combined with a subkey bit. The gray parts are the eliminated intermediate S-boxes and states. In the left-side of each S-box the amount of its computations is shown. The influences of subkeys on some intermediate nibbles are also shown in braces.

In combination with some other ideas like early abort [6], sieve in the middle [7] or indirect partial matching [3] techniques, the matching part of the generalized MITM attack can be more improved. Here, the focus is on partial matching and it is assumed that the attacker will use these ideas after choosing the proper condition (i.e. the cut-set and the matching variable) for the attack.

C. GENERALIZED MITM ATTACK DETAILS

The approach of generalized MITM attack is to automatically obtain the best forward and backward key bits during the attack procedure, whereas in basic MITM or biclique attack, the attacker tries to partition the master key at the beginning. That's why the initial selection step in generalized MITM attack doesn't contain the key partitioning like biclique attack. Clarifying how to perform pre-computations and recomputations in step 3, how to obtain the forward and backward key bits, and also how to execute key recovery in step 4 of generalized MITM attack are the aim of this section.

Suppose that both cut-sets of $\mathcal{K}$ and $\mathcal{C}$, and also the matching variable $\mathcal{V}$ are chosen and fixed. Also, all extra computations are removed in pre-processing step. Now, attacker wants to perform the step 3. Therefore, she does the pre-computations by an arbitrary chosen value for n-bit master key $K = \{k_1, k_2, \ldots, k_n\}$ (which is equivalent to cut-set $\mathcal{K}$). After that, the recomputations should be performed operation by operation (e.g. S-box, XOR, etc.). Starting from $\mathcal{C}$ to calculate matching variable $\mathcal{V}$, attacker will face various operations (no matter what the direction is). For each operation, assume that:

- all the input values are pre-computed and stored;
- all bits of master key K which have influenced on the inputs are known ($K^{in} = \{k_{i_1}, k_{i_2}, \ldots, k_{i_a}\}$);
- there are $K^{o_1} = \{k_{j_1}, k_{j_2}, \ldots, k_{j_b}\}$ bits of master key K which have influenced on inside, and also computational complexity of the operation;

- there are $K^{o_2} = \{k_{l_1}, k_{l_2}, \ldots, k_{l_c}\}$ bits of master key K which have influenced on inside, but not computational complexity of the operation.

Then, computational complexity of the operation is equal to $2^{\#(K^{in} \cup K^{o_1})}$ (in which $\cup$ means union of the two set), and all bits of master key K which have influenced on the output of the operation are $K^{out} = (K^{in} \cup K^{o_1} \cup K^{o_2})$. Afterwards, all the $2^{\#K^{out}}$ output values along with values of K^{out} bits should be stored as pre-computations for the other operations. This procedure continues until the matching variable $\mathcal{V}$ will be obtained. Now, $K^{op} = (K^{in} \cup K^{o_1})$ for each active operation op from cut-set $\mathcal{C}$ to matching variable $\mathcal{V}$ in both forward and backward direction has been found. By defining all active non-linear operations as set $\mathcal{OP}$, and total number of non-linear operations of the targeted block cipher as N, the normalized time complexity of the attack is equal to the following equation:

$$C_{time} = \frac{\Sigma_{op_i \in \mathcal{OP}} \{2^{\#K^{op_i}}\}}{N} \tag{1}$$

In the calculation of C_{time}, the operations with higher $m \doteq \#K^{op}$ are dominant. In order to reduce memory complexity, the pre-computations can be started from exactly before these dominant operations. Therefore, it is necessary to choose a proper M to divide the operations into dominant (with $m \geq M$) and non-dominant (with $m < M$) segments. The proper M has negligible effect on time complexity while it leads to huge reduction in memory complexity.

Let $\mathcal{OP}^f_m$ and $\mathcal{OP}^b_m$ be sets of operations with $\#K^{op} = m$ in forward and backward direction, respectively (i.e. $\mathcal{OP}^f_m = \{op \in \mathcal{OP} | 1 \leq m \leq n \& \#K^{op} = m \& op$ is in forward direction$\}$; respectively $\mathcal{OP}^b_m$ for backward direction). In addition, let K^{op*} be the complementary form of K^{op} (i.e. $K^{op*} = K - K^{op}$). Then, forward (K^f) and backward (K^b) key bits for generalized MITM attack can be extracted according to the below:

- $K^f = \cup_{op \in OP^f_{M-1}} \{K^{op*}\}$
- $K^b = \cup_{op \in OP^b_{M-1}} \{K^{op*}\}$

Normally but not necessary, M should be equal to n or $n - l$ (in which l is the smallest word size of the block cipher) to obtain the best result. When OP^f_{M-1} (resp. OP^b_{M-1}) is empty, then the non-empty set of OP^f_m (resp. OP^b_m) with greatest m and $m < M$ should be considered. After obtaining K^f and K^b, key partitioning into four segment will be a straightforward procedure as follows:

- Group bits: $K^G = K - \{K^f \cup K^b\}$
- Common key bits: $K^C = K^f \cap K^b$
- Forward key bits: $K^F = K^f - K^C$
- Backward key bits: $K^B = K^b - K^C$

When generalized MITM attack can turn into a biclique attack (as it is mentioned in the next section), then the key partitioning for biclique attack will be $K^g = K^G \cup K^C$, $K^f = K^B$, and $K^b = K^F$. It should be mentioned that as in the biclique attack, forward and backward keys are considered in the biclique part (and not in the matching part), the definitions of forward and backward keys are reverse for biclique attack in comparison to generalized MITM attack.

When the master key bits are partitioned into four segments of $K = (K^G, K^C, K^F, K^B)$, an attacker can extract the correct master key as below:

- Perform the followings for all possible values of (K^G, K^C) bits.
- For each value of K^B bits, perform partial encryption from cut-set $\mathcal{C}$ to obtain matching variable $\mathcal{V}$ in forward direction by using pre-computation and recomputation technique on K^F bits.
- For each value of K^F bits, perform partial decryption from cut-set $\mathcal{C}$ to obtain matching variable $\mathcal{V}$ in backward direction by using pre-computation and recomputation technique on K^B bits.
- Compare the values of $\mathcal{V}$ in forward and backward direction for a fixed values of K^F and K^B to filter out the wrong keys.
- Exhaustive search on the remaining keys to obtain the correct master key.

The memory complexity of the attack is equal to $2^{\#(K^F, K^B)-1}$ as the pre-computations in forward and backward directions should be done until the last part of K^F and K^B, respectively. It should be noticed that if the smallest word-size of the target block cipher is bigger than a bit, all the above mentioned in this section can be repeated with replacing bit by word and also some additional slight modifications. Consideration of words instead of bits can dramatically speed up the whole procedure.

III. COMPARISON WITH CONVENTIONAL VERSIONS OF MITM ATTACKS

In this section, it will be discussed that the generalized MITM attack can cover some previous improved versions of MITM attacks and also provides some new cases, by the following claims.

Claim 1: Any basic MITM, 3-subset MITM, splice-and-cut, or biclique attack is also a generalized MITM attack.

For the sake of the mentioned claim, some descriptions for each attack are provided one by one as follows (see Figure 3 for more details):

- Basic MITM attack: let $\mathcal{C}$ be equal to P (or equivalently C) and $\mathcal{V}$ equal to the matching variable of the basic MITM attack. Now, the generalized MITM attack and the basic MITM attack are exactly the same.
- 3-subset MITM attack: generally, the 3-subset MITM attack removes restrictions on the choice of key bits compared to the basic MITM attack. Therefore, there is no change in the main procedure, and the description provided for the basic MITM attack works for the 3-subset MITM attack too.
- Splice-and-cut attack: considering the internal state of splice-and-cut as cut-set $\mathcal{C}$, the remaining parts of both

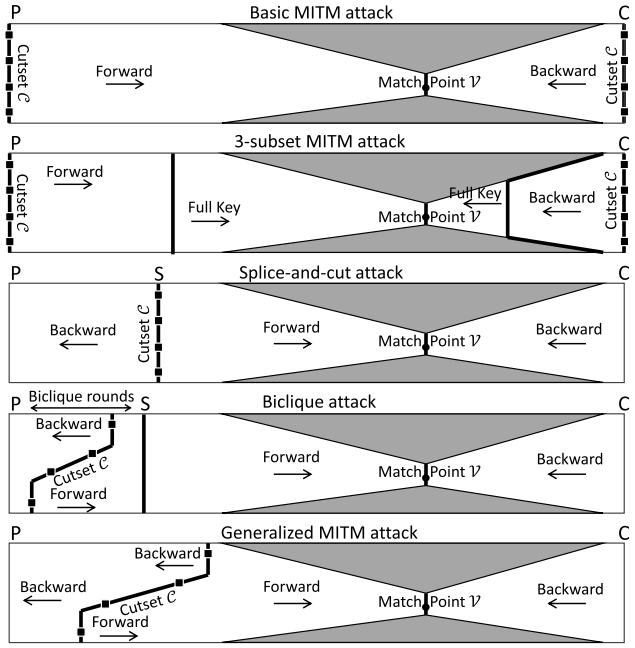


FIGURE 3. Comparison of basic MITM, 3-subset MITM, splice-and-cut, biclique, and generalized MITM attacks.

splice-and-cut and generalized MITM attack are the same.

- Biclique attack: according to [17] for any biclique attack, a cut-set, namely root cut-set, can be found in the biclique part which has not been affected by both forward and backward differentials. Considering that cut-set equal to $\mathcal{C}$ and similar matching variable for a generalized MITM attack as the biclique attack, then the biclique attack can be completely the same as the generalized MITM attack.

Claim 2: There are some cases which can be covered by the generalized MITM attack, but not by the previous MITM attacks

It must be shown that not any generalized MITM attack can be presented by one of the previous MITM attacks. Choosing the cut-set $\mathcal{C}$ in a way that its variables be in more than one internal state leads to a new case which cannot be covered by basic MITM, 3-subset MITM, and splice-and-cut attacks. Considering such a cut-set, the generalized MITM attack can be applied with any cut-set as the starting state regardless of its position, while in biclique attack as the biclique part is always in the first or last rounds of the cipher, the root cut-set of biclique cannot be anywhere. In addition, the biclique is always between two round states, but using generalized MITM attack, this condition is removed and actually the biclique can be between two cut-sets. Therefore, generalized MITM attack can cover some new cases in comparison to the previous MITM attack variants (see Figure 3).

It can be concluded from the above claims (III and III) that providing security against generalized MITM attack ensures the security against the other mentioned MITM attacks. Also it should be mentioned that the generalized MITM attack does not rely on what the method of wrong keys filtering in the matching part is. Hence, it is clearly possible to use any idea such as early abort, sieve in the middle or indirect partial matching techniques instead of simple matching variable check to have more effective attacks.

IV. AUTOMATION OF GENERALIZED MITM ATTACK

Generalized MITM attack provides a lot of cases which should be investigated to find the best one. Clearly, the total number of possible generalized MITM attacks is more than the ability of a cryptanalyst to check them all manually. Hence, an automated method for searching around all these attacks and providing the best one is required. In this section, the goal is to introduce the exact search algorithms required for this automation. So, first a method for defining a block cipher with emphasis on its diffusion property is provided in the following sub-section, and then algorithms required for total procedure of generalized MITM attack are presented.

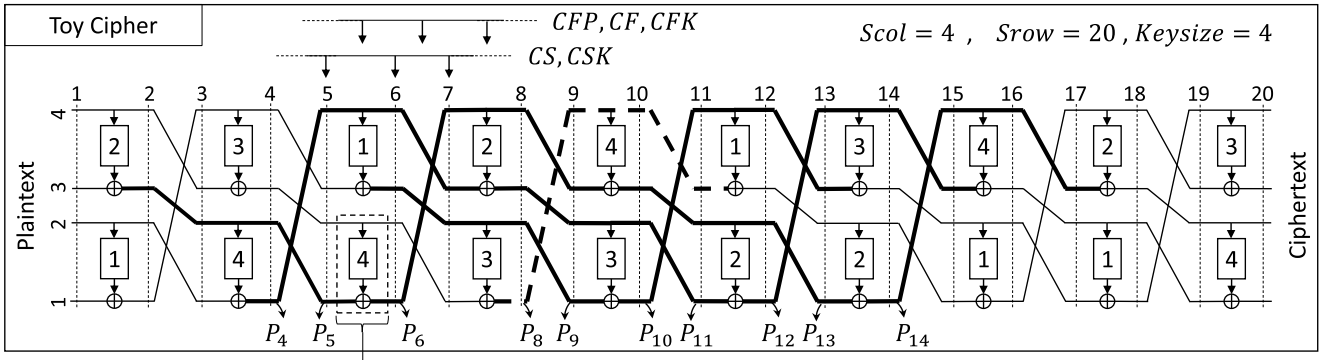
A. BLOCK CIPHER DIFFUSION

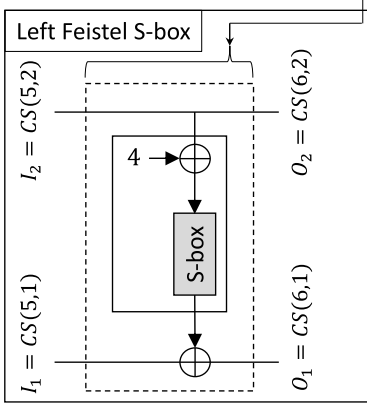
Diffusion of block cipher and key schedule algorithms has a key role on measuring the effectiveness of generalized MITM attack. Therefore, providing a proper model for analysis of diffusion in these algorithms shed light on how to automate the total procedure of generalized MITM attack. Any block cipher (or key schedule) algorithm can be considered as a set of internal states along with relations between these internal states. More precisely, suppose that a block cipher algorithm consists of *srow* internal states, each with *scol* words, and also its corresponding key schedule has *krow* internal states, each with *keysize* words (the variables are shown in the *italic* form; *scol* and *keysize* are equivalent to block and key sizes, respectively). In the diffusion perspective, one should know which of the previous or next internal state words are required to compute a current state word in forward and backward direction, respectively. Hence, the following knowledges about relations between current internal state words and previous/next internal state words should be determined:

- Properties of the operations (or intermediate round functions) between internal states in forward and backward directions
- Direct effect of chosen master key bits on performance of each operation in forward and backward direction (only for block cipher algorithm)

These knowledge can be accurately obtained and stored in the following arrays:

1) $CFP(1 : srow - 1, 1 : scol, 1 : 9)$: a 3-dimensional $(srow - 1) \times scol \times 9$ array to show the place of each operation and its nine parameters; consists of length (greater than 0 for each operation), total complexity (between 0 and 1; 0 for linear operations and 1 for non-linear operations), subkey words (according to values of this parameter, the subkey words are chosen from the current internal state of the key schedule), update





CFK values for the Left Feistel S-box	
K_1	$CFK(5,1,1{:}4) = \{0,0,0,1\}$
K_2	$CFK(5,2,1{:}4) = \{0,0,0,0\}$

CF values for the Left Feistel S-box	
$\begin{bmatrix} O_1 \\ O_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$	$CF(5,1{:}2,1{:}2,1) = \begin{Bmatrix} 1 & 1 \\ 0 & 1 \end{Bmatrix}$
$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} O_1 \\ O_2 \end{bmatrix}$	$CF(5,1{:}2,1{:}2,2) = \begin{Bmatrix} 1 & 1 \\ 0 & 1 \end{Bmatrix}$
$\begin{bmatrix} O_1 \\ O_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}$	$CF(5,1{:}2,1{:}2,3) = \begin{Bmatrix} 1 & 0 \\ 0 & 0 \end{Bmatrix}$
$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} K_1 \\ K_2 \end{bmatrix}$	$CF(5,1{:}2,1{:}2,4) = \begin{Bmatrix} 1 & 0 \\ 0 & 0 \end{Bmatrix}$

CFP values for the Left Feistel S-box	
Length	$CFP(5,1,1) = 2$
Total Complexity	$CFP(5,1,2) = 1$
Subkey words	$CFP(5,1{:}2,3) = \{4,0\}$
Update flag	$CFP(5,1,4) = 0$
Subkey effect on complexity	$CFP(5,1{:}2,5) = \{1,0\}$
Subkey effect on outputs	$CFP(5,1{:}2,6) = \{1,0\}$
Subkey effect on inputs	$CFP(5,1{:}2,7) = \{1,0\}$
Inputs effect on complexity	$CFP(5,1{:}2,8) = \{0,1\}$
Outputs effect on complexity	$CFP(5,1{:}2,9) = \{0,1\}$

FIGURE 4. Toy block cipher and Left Feistel S-box reconsideration.

flag (0 means the selected internal state of the key schedule must not be updated; 1 means it must be updated), effects of subkey words on computational complexity (between 0 and 1), computational complexity of input/output words calculation (between 0 and 1), and calculation effects of input/output words on computational complexity (between 0 and 1). As an example, *CFP* array elemets for a Left Feistel S-box operation are shown in the Figure 4. This operation is placed between 5^{th} and 6^{th} internal state and it has two-word length started from word 1. Therefore, $CFP(5, 1, 1)$ should be equal to 2 which means that in the corresponding place, there is an operation with length 2. Also, the total complexity of this operation is equal to 1 S-box operation. Subkey words are $\{4, 0\}$ which means that the first subkey word is 4^{th} word of current internal state of key schedule and the second one is empty (or equivalently, this subkey has only one word). Update flag is 0, because Toy block cipher in Figure 4 has not key schedule. The one-word corresponding subkey directly affects on S-box calculation. Therefore, $CFP(5, 1 : 2, 5)$ should be equal to $\{1, 0\}$. Also, the subkey only affects on the left word of output and input in both forward and backward direction, respectively. So, $CFP(5, 1 : 2, 6)$ and $CFP(5, 1 : 2, 7)$ are both equal to $\{1, 0\}$. In addition, the left input and output words of the operation have not effect on S-box calculation, while the right input and output words can directly activate the S-box. Therefore, $CFP(5, 1 : 2, 8)$ and $CFP(5, 1 : 2, 9)$ should be both equal to $\{0, 1\}$.

2) $CF(1 : srow - 1, 1 : scol, 1 : scol, 1 : 4)$: Four 3-dimensional $(srow - 1) \times scol \times scol$ boolean arrays to show the effects of input words on output words of an operation and vice versa, and also effects of subkey words on output and input words of that operation, respectively. In fact, *CF* simulate the diffusion property of each operation in block cipher. For example, when the second word of i^{th} internal state has affect on the first word of $(i + 1)^{th}$ internal state in forward direction, then $CF(i, 1, 2, 1)$ is equal to 1. In Figure 4, these four arrays for a two-word Left Feistel S-box are shown.
3) Similar arrays with CFP and CF for key schedule, namely *KFP* and *KF*, respectively. For *KFP* array, the elements related to subkey words, update flag and subkey effects are all zero.
4) $CFK(1 : srow - 1, 1 : scol, 1 : keysize)$: a 3- dimensional $(srow - 1) \times scol \times keysize$ boolean array to show the master key words which have influenced on the operations of the cipher. More precisely, the place of subkey words in the current internal state of key schedule for each operation is stored by $CFP(i, j, 3)$ in which $1 \leq i \leq srow - 1, 1 \leq j \leq scol$, and the set of master key words which have influence on each subkey word is provided by

$CFK(i, j : j + CFP(i, j, 1) - 1, 1 : keysize)$. In Figure 4, with $i = 5, j = 1$, $CFP(5, 1, 1) = 2$ and $keysize = 4$, the value of $CFK(5, 1 : 2, 1 : 4)$ for a Left Feistel S-box is shown.

5) $K\_Marked(1 : krow)$: a boolean array to show which internal states of the key schedule must be used for extracting the subkeys. Hence, CFK is calculated from key schedule internal states (i.e. KSK in the following) with update flags and K_Marked. At the beginning, the first internal state of key schedule with true K_Marked is considered as the current state for key schedule. Then CFK for each operation should be obtained from this state until the update flag of an operation will be true. In this situation, before obtaining CFK for the next operation, the next internal state of key schedule with true K_Marked should be considered. This procedure will continue until all operations obtain their subkeys. More precisely, it can be said that $CFK(i, j : j + CFP(i, j, 1) - 1, 1 : keysize)$ is exactly equal to $KSK(selected\_Krow, CFP(i, j : j + CFP(i, j, 1) - 1, 3), 1 : keysize)$ in which the $selected\_Krow$ should be obtained by consideration of update flags (i.e. $CFP(., ., 4)$) and K_Marked.

The above knowledge are important to define a block cipher for the automation of generalized MITM attack. However, some additional knowledge are required to perform the attack. These knowledges which should have initial values and be updated during the attack procedure, are as below:

- The active words in each internal states
- Effect of each master key word on each internal state word
- Direction of activation for each operation

They can be defined and stored accurately as follows:

1) $CS(1 : srow, 1 : scol)$: a 2-dimensional $srow \times scol$ array to show the situation of all the internal state words of block cipher algorithm or equivalently input/output words of each operation (0 means it is not activated, 1 means it is activated, and -1 means it is eliminated). The initial values for whole of CS array elements are 0, and at the end of generalized MITM attack, all elements should be 1 or -1. In Figure 4, CS elements for a Left Feistel S-box are shown.
2) $CSK(1 : srow, 1 : scol, 1 : keysize)$: a 3-dimensional $srow \times scol \times keysize$ boolean array to show which master key words (from 1 to $keysize$) have influenced on the cipher intermediate states. Again, the initial values for all CSK array elements are 0 and they will be updated during the attack procedure.
3) $CDir(1 : srow - 1, 1 : scol)$: a 2-dimensional $(srow - 1) \times scol$ boolean array to indicate activation direction for each operation (1 for forward, -1 for backward, and 0 for inactivation or initial value)
4) Similar arrays with CS, CSK and $CDir$ for key schedule, namely KS, KSK and $KDir$, respectively.

To summarize the procedure of a generalized MITM attack with the above arrays, the following steps should be performed:

0) Block cipher definition:
   a) Provide CFP, CF, KFP, KF and K_Marked for a block cipher.
   b) Provide CS, CSK, KS, KSK, $CDir$ and $KDir$ with initial values.
1) Initial selection:
   a) Choose a proper cut-set $\mathcal{K}$ from key schedule as the master key and calculate KS, KSK, $KDir$ and CFK.
   b) Choose a proper cut-set $\mathcal{C}$ from intermediate states of the block cipher and set the corresponding elements in CS equal to 1.
   c) Choose a proper matching variable $\mathcal{V}$.
2) Pre-processing:
   a) Remove extra intermediate states of partial matching in CS by setting value of -1 for each corresponding word.
3) $\mathcal{C}$ to $\mathcal{V}$ propagation:
   a) Calculate all elements of CS and CSK in forward and backward directions.
   b) Calculate attack complexities with the final states of these arrays.
4) Checking the guessed keys: similar to Section II-C.

In the following section, the automation method for generalized MITM attack on a target block cipher by the mentioned arrays is provided.

B. ALGORITHMS FOR TOTAL PROCEDURE

In order to automate the generalized MITM attack, the following steps should be considered, respectively:

Step 1 (Initial Selection):

- Providing the notion of point and independent points in block ciphers.
- Introducing the approach of constructing cut-sets by independent points.
- Proposing Exhaustive Recursive Search (ERS) to find cut-sets of a block cipher.
- Choosing a point as the matching variable $\mathcal{V}$.

Step 2 (Pre-Processing):

- Presenting Elimination Algorithm to remove all the extra computations in the matching part.

Step 3 ($\mathcal{C}$ to $\mathcal{V}$ Propagation):

- Providing the propagation algorithm to obtain the effects of master key words on each intermediate variable and operation between $\mathcal{C}$ and $\mathcal{V}$.
- Presenting an algorithm for calculating the complexities (time, memory and data).

It should be emphasized that the target of automation is calculating the attack complexities, not implementing the attack. Hence, the last step of generazlied MITM attack

(i.e. checking the guessed keys) is not necessary for automation. Of course, an attacker can utilize the corresponding parameters of the best attack to perform the key recovery according to Section II-C.

1) INITIAL SELECTION

Considering all of the smallest intermediate variables (or words) for finding cut-sets of a block cipher makes extra complexity and redundancy, while it has no gain for this goal. For example, input and output word of a permutation have the same property, and there is no difference between them to select for a cut-set. Therefore, notion of point is provided to remove these extra cases and present unique answers, as follows:

Definition 2 (Point): A set of all the smallest intermediate variables or words (i.e. indexes of CS) with the exact same values in the cipher is called a point. Therefore, all the indexes in a point have always the same CS and CSK values.

As an example, $I_2 = CS(5, 2)$ and $O_2 = CS(6, 2)$ are in one point in Figure 4, while $I_1 = CS(5, 1)$ and $O_1 = CS(6, 1)$ are in different points. It can be seen that a point is a union of words, each can be obtained from the others by at most a permutation. The toy cipher which is shown in Figure 4 has the following 20 points:

$$\begin{aligned} P_{2i-1} &= (CS(2i-1,1),\ CS(2i-2,2),\ CS(2i-3,2),\\ &\quad CS(2i-4,3)), \quad 1 \le i \le 10;\\ P_{2i} &= (CS(2i,1),\ CS(2i+1,4),\ CS(2i+2,4),\\ &\quad CS(2i+3,3)), \quad 1 \le i \le 10. \end{aligned}$$

By reminding Definition 1, if any two bits (or equivalently two words) of a cut-set be on one point, then they can be calculated from each other. This has contradiction with the notion of cut-set which choose the lowest possible words in order to perform encryption/decryption assumed that the master key is given. Therefore, any two words of a chosen cut-set must be in different points. Hence, without loss of generality, it can be said that any cut-set with b words can be shown by b distinctive points. This opens new insights in obtaining cut-sets of a cipher. Firstly, it omits many identical cut-sets which include the same points but different intermediate variables. Secondly, it shows a determinestic method for acceptance or rejection of any given set of b distinctive points as a cut-set. More precisely, if all the given b points are independet from each other, then they must construct a cut-set. Independency of two points is defined as follows:

Definition 3 (Independency of Two Points P and Q): P and Q are independent points in a block cipher, if and only if $Q \in$ Independent(P), which is described below:

1. *In forward (resp. backward) direction, find all variables (not points) required for calculating P by $CF(.)$ and mark them all.*
2. *Repeat marking for newly marked variables until the ciphertext (resp. plaintext);*
3. *Return the smallest set of points containing all the unmarked variables as Independent(P)*

It can be easily checked that if $Q \in$ Independent(P) then $P \in$ Independent(Q). As an example of independent points, consider point P_8 in the Toy cipher of Figure 4, then Independent(P_8) $= \{P_4, P_5, P_6, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}\}$. With considering more than two points, the independency of these points can be interpreted as independency of all chosen pairs of them. Therefore, in order to find all cut-sets of a block cipher, one method can be finding of all possible b independent points in it. This is done by a proposed recursive algorithm (namely ERS) starting from basic input points (for example all points involved in block cipher) which is provided in Algorithm 1. The main procedure of ERS is as follows:

1. Choose a new point P from a set of *InputPoints*;
2. Add P to a set of points $\mathcal{C}$;
3. If $\mathcal{C}$ has b member, then provide $\mathcal{C}$ as a cut-set, and remove P from it;
4. Else set *InputPoints* = Independent(P) and repeat the procedure.

Algorithm 1 can be more accelerated with some middle checks. These checks come from the inherent fact that all common points of a chosen cut-set with input or output points of an intermediate function, must not breach the independency condition for that intermediate function. Therefore, during the search for cut-sets, this independency condition can be checked for all relevant intermediate functions and wrong cut-sets can be dropped more quickly. This checking algorithm which should be used in the place of optional checks in ERS algorithm (i.e. Algorithm 1), is provided in Algorithm 2.

Hence, cut-sets of a determined block cipher can be found by combination of Algorithms 1 and 2. In addition, as the matching variable $\mathcal{V}$ description can also be modified to the matching point $P_{\mathcal{V}}$, all possible matching points are equivalent to all points of the block cipher.

These proposed algorithms can also be performed on key schedule to extract its cut-sets. Clearly, any cut-set of key schedule can be considered as the master key words. Then, according to this chosen master key and key schedule algorithm, the effects of master key on subkey words should be found and inserted into the block cipher.

2) PRE-PROCESSING

The elimination procedure for removing extra intermediate variables and functions of partial matching can be provided as follows:

- In forward (resp. backward) direction, find all **points** required for calculating the $P_{\mathcal{V}}$ by $CF(.)$ and mark them all.
- Repeat marking for newly marked **points** until the ciphertext (resp. plaintext);
- Return all the unmarked points as Elimination($P_{\mathcal{V}}$)

Hence, Elimination($P_{\mathcal{V}}$) indicates the points which should be removed. Considering a cut-set as b independent points, it can be easily checked that there are only two cut-sets

Algorithm 1 ERS Algorithm

Find all points of the cipher with top-down and left-right order and save them into *Points*
Set a global array $CS(1:srow, 1:scol) = 0$
Set global variables $cutsetnumber = 0$ and *cutsets* empty
Call $cutset = \text{ERS}(Points, 1, empty)$
****
procedure $cutset$=ERS($Points, value, cutset$)
    Set $m = 0$
    Set *Rec* empty
    while $m < Length(Points)$ **do**
        Set $m = m + 1$ and then $P = Points(m)$
        Set $CS(P) = value$ ▷ All $CS(i, j)$ with $\{i, j\} \in P$ must be set to *value*.
        Set all $\{Q|Q \in Points, Q \notin \text{Independent}(P), CS(Q) = 0\}$ to $value + scol$
        Add P to *cutset*
        Do an optional check for extracting the *flag* (the default *flag* is zero)
        if $(flag == 0)$ **then**
            if $(value < scol)$ **then**
                Set $NewPoints = \text{Independent}(P)$
                if $(\text{Length}(NewPoints) >= scol - value)$ **then**
                    $cutset = \text{ERS}(NewPoints, value + 1, cutset)$
                end if
            else
                Set $cutsetnumber = cutsetnumber + 1$
                Add *cutset* into *cutsets*
            end if
        end if
        Remove P from *cutset*
        $\forall\{Q|CS(Q) = value + scol\}$, set CS(Q) back to 0
        Add P to *Rec*
        Set $CS(P) = -value$
        Repeat the optional check to extract a new *flag* (the default *flag* is zero)
        if $(flag == 1)$ **then**
            $m = Length(Points)$
        end if
    end while
    $\forall X \in Rec$ Set $CS(X) = 0$
end procedure

constructed by unremoved points containing matching point $P_{\mathcal{V}}$, namely $\mathcal{C}_P$ and $\mathcal{C}_C$, in the plaintext-side and ciphertext-side of $P_{\mathcal{V}}$, respectively (see matching part of Figure 2 as an example). These cut-sets will be used in the propagation procedure.

3) $\mathcal{C}$ TO $\mathcal{V}$ PROPAGATION

The procedure of finding the number of effective master key words on each intermediate function and variable, starting from a chosen cut-set $\mathcal{C}$ to another arbitrary cut-set $\mathcal{C}_2$ in forward or backward direction, is called propagation. After choosing a matching $P_{\mathcal{V}}$, two cut-sets of $\mathcal{C}_P$ and $\mathcal{C}_C$ are obtained. Hence, $\mathcal{C}$ to $\mathcal{V}$ propagation refer to union of propagations from $\mathcal{C}$ to $\mathcal{C}_P$ and $\mathcal{C}_C$, in forward and backward direction, respectively. It is a direct procedure with the simple algorithm as Algorithm 3 for forward direction. Though Algorithm 3 is presented for a block cipher algorithm in forward direction, it can also be used directly on the key schedule (of course, there is not any match point in the key schedule). For the backward direction, the general procedure is similar to forward direction.

If Algorithm 3 is performed on the key schedule, it must not continue after reaching the first or last key schedule state, while for performing on the cipher, it should continue until all intermediate functions/operations in foward (or backward) direction are considered. After performing propagation algorithm, all the arrays are ready to compute the time complexity which is presented in Algorithm 4.

It should be mentioned that the time complexity of the key schedule in Algorithm 4 can be obtained with the same procedure of Algorithm 4 but on the key schedule. In addition, the lower bound for data complexity can be achieved by

Algorithm 2 Checking Algorithm

Consider $CS(1:srow, 1:scol)$ as a global array
procedure $flag = \text{check}(Point, Dir, flag)$
    $flag = 0$
    Find the next intermediate function F includes the *Point* in *Dir* direction
    Set A as all outputs of F which require the *Point* to be calculated in *Dir* direction
    Set $Xpass = 0$, and $Xin = 0$
    for $x \in A$ **do**
        Set B as all the inputs of F which have effect on x in *Dir* direction
        Set $X0 =$ number of B input elements with values of greater equal than zero
        if $X0 == length(B)$ **then**
            Set $Xpass = Xpass + 1$
        end if
    end for
    Set C as all inputs of F which have effect on at least one member of A
    Set $Xin =$ number of C input elements with values of greater equal than zero
    if ($Xin > Xpass$ && C has at least one input element with positive value) **then**
        $flag$=1
    end if
end procedure

Algorithm 3 Propagation in Forward Direction

Global inputs: (CS, CSK, CF, CFP, CFK), $srow$, $scol$, $keysize$, cut-set $\mathcal{C}$, $P_{\mathcal{V}}$
Assumed that $CS(\text{Elimination}(P_{\mathcal{V}})) = -1$
Assumed that $CS(\mathcal{C}) = 1$
for $i = 1$ to $srow - 1$ **do**
    Set $j = 1$
    while $j <= scol$ **do**
        if $CS(i, j) == 1$ **then**
            Set $A =$ every $CS(i+1, .)$ index which is influenced by $CS(i, j)$
            for $a \in A$ which $CS(i+1, a) == 0$ **do**
                if all required $CS(i, .)$ for calculation of $CS(i+1, a)$ are equal to 1 **then**
                    Set $CS(P) = 1$ in which $CS(i+1, a) \in point\ P$
                    Set $CSK(P, .)$ according to CF, CFK, and CSK
                end if
            end for
        end if
        if All the *inputs* of the intermediate function is 1 **then**
            Set $CDir$ of the intermediate function in $CFP(.)$ (1 for forward direction)
            $j =$ end of internal function
        end if
        Set $j = j + 1$
    end while
end for
if $CS(ciphertext) == 1$ && $CS(plaintext) == 0$ **then**
    Set $CS(plaintext) = 1$
    Repeat the above procedure again
end if

the minimum number of activated plaintext/ciphertext words when all the master key words are considered one by one (see Figure 2 as an example). There is an additional input in Algorithm 4, namely *Vector*. It is leveraged for separation of pre-determined key words from unknown ones. So, it can be used for not considering some pre-determined key words in the attack complexity with the assumption that they should be guessed before. These pre-determined key words can be group or common key bits, defined in Section II-C. Hence, by arrangement of *Vector* for some key words with fewest

Algorithm 4 Computational Complexity Calculation

Global inputs: CS, CSK, CF, CFP, CFK, KS, KSK, $Usize$, $keysize$, $Vector$, CK ▷ $Usize$ is the word size; $Vector$ is a boolean array with $keysize$ length; CK is complexity of the key schedule.
$CC = 0$, $SBOX = 0$, $NZ = \Sigma(Vector - 1) \times (-1)$
for each intermediate function IF according to CFP **do**
    Get $CDir$, UC, and TC from CFP ▷ Unit and Total Complexities of the IF
    if Dir is not 0 **then** ▷ It means if the IF is activated.
        $SBOX = SBOX + TC$
        $W = min(TC, \Sigma_{i=1}^{Len} UC(Dir))$
        $Keys =$ all key words involved in IF computations
        $SUM = Keys \odot Vector$ ▷ Inner product
        $CC = CC + W \times 2^{Usize \times (SUM + NZ)}$
    end if
end for
$Complexity = log_2((CK + CC)/SBOX)$

Algorithm 5 Fully Automated Generalized MITM Attack

Global inputs: CP, CF, CFK, KS, KSK
Find all points of the cipher and store them in *Points*
Choose a random cut-set in key schedule as the main key bits and calculate all the subkeys with performing the propagation algorithm on the key schedule
Find all cut-sets of the cipher according to Algorithms 1 and 2 and store them in *cutsets*
for $P_{\mathcal{V}} \in Points$ **do**
    Set $extra\_points =$ Elimination($P_{\mathcal{V}}$)
    for $C \in cutsets$ **do**
        if C has not any common points with $extra\_points$ and $P_{\mathcal{V}}$ **then**
            Set $CS(.) = 0$
            Set $CS(extra\_points) = -1$
            Set $CS(C) = 1$
            Propagate C in both directions until $P_{\mathcal{V}}$ using Algorithm 3
            Calculate computational complexity by Algorithm 4
            Save the results
        end if
    end for
end for

S-box activation in forward and backward direction between cut-set $\mathcal{C}$ and point $P_{\mathcal{V}}$, the attack time and data complexities can be exactly re-calculated while the memory complexity of the attack can also be obtained from total number of chosen forward and backward key words minus one. For purpose of finding lower time complexity bound, *Vector* should be set as a total one array with *keysize* length.

4) AUTOMATION PROCEDURE

According to the all above mentioned algorithms, the generalized MITM attack can be proposed by a fully automated procedure as Algorithm 5.

V. EXPERIMENTAL RESULTS

In this section, the results of automated generalized MITM attack on various block ciphers are presented. The targeted ciphers are HIGHT [18] and Piccolo-128 [19] as Feistel block ciphers, and CRAFT [20], and AES-128 [21] as SPN block ciphers. The block size of HIGHT, Piccolo-128 and CRAFT are 64-bit length, while AES-128 has 128-bit block length. On the other hand, HIGHT, Piccolo-128 and AES-128 have byte-wise calculations (8-bit word size), while CRAFT has nibble-wise calculations (4-bit word size). All the block ciphers have a 128-bit master key shown by $K = (K^0, K^1, \ldots, K^{i-1})$ in which $i = \frac{128}{wordsize} - 1$. More details of these block ciphers including key schedules are provided in [18]–[21], and, the overall schematics along with internal layers of them are depicted in Figure 5.

A. HIGHT

1) GENERALIZED MITM ATTACK ON HIGHT

Applying generalized MITM attack on full HIGHT results in the followings:

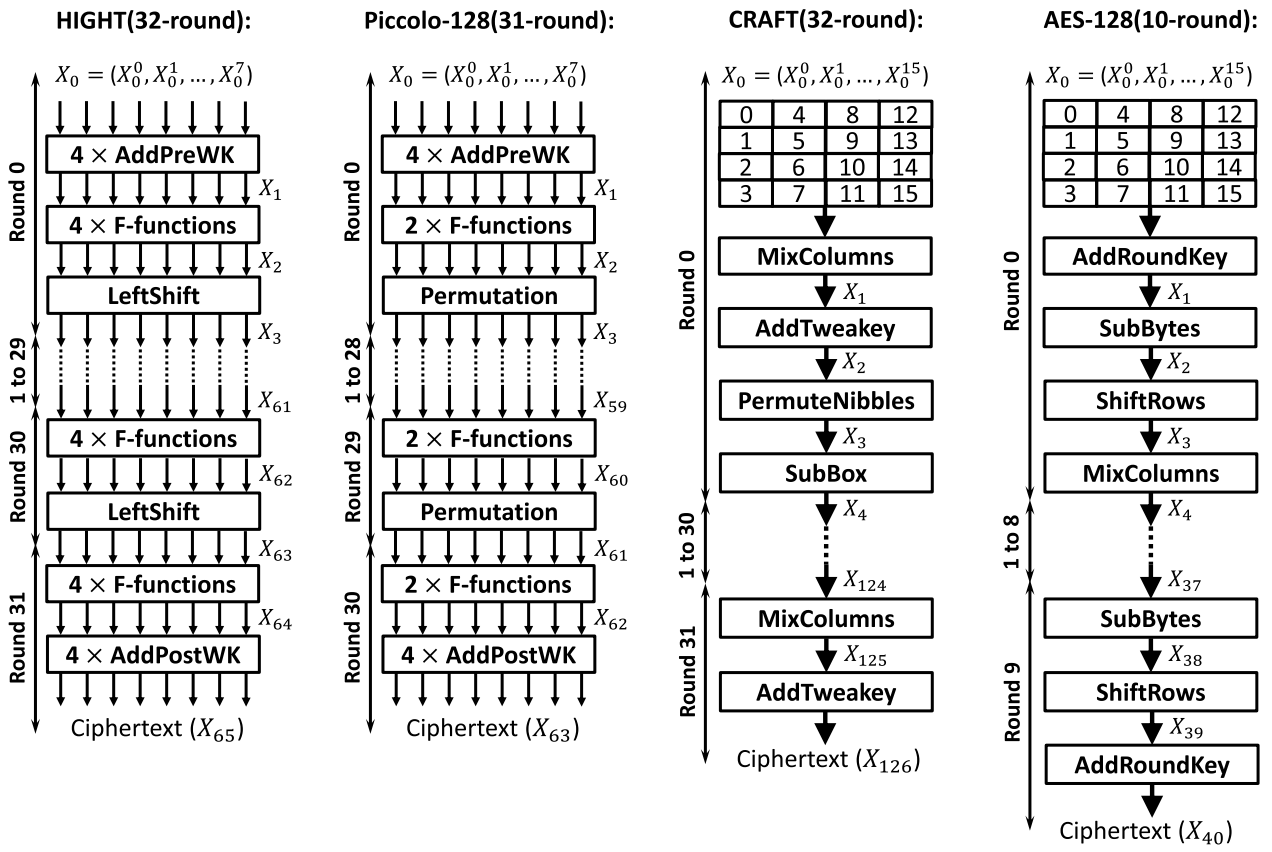


FIGURE 5. Schematics and internal layers of HIGHT, Piccolo-128, CRAFT and AES-128.

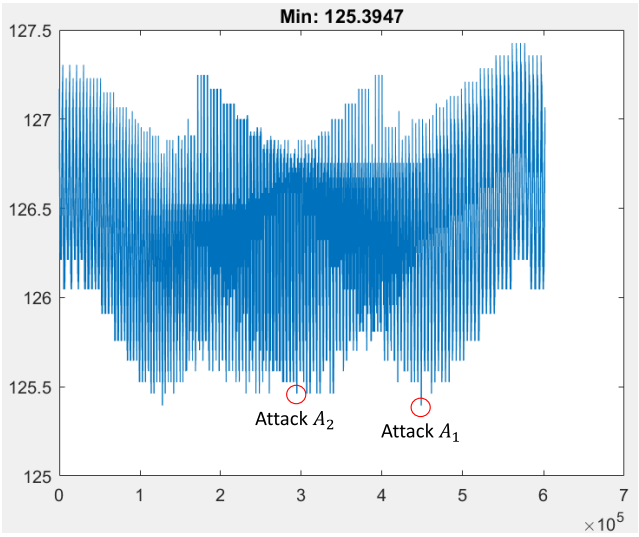


FIGURE 6. Time complexity of all generalized MITM attacks on HIGHT; vertical axis shows logarithm of the time complexity, and horizontal axis shows the attack number.

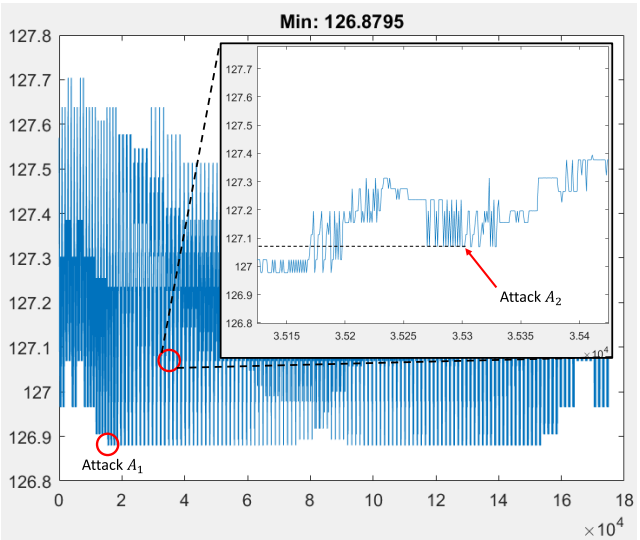


FIGURE 7. Time complexity of all generalized MITM attacks on Piccolo-128; vertical axis shows logarithm of the time complexity, and horizontal axis shows the attack number.

- According to its simple key schedule, any random chosen cut-set in the key schedule has the same results. Therefore, without loss of generality, the master key is chosen for that cut-set.
- There are 144 points, 6162 cut-sets, and 601928 possible attacks (considering all acceptable combination of matching points and cut-sets).

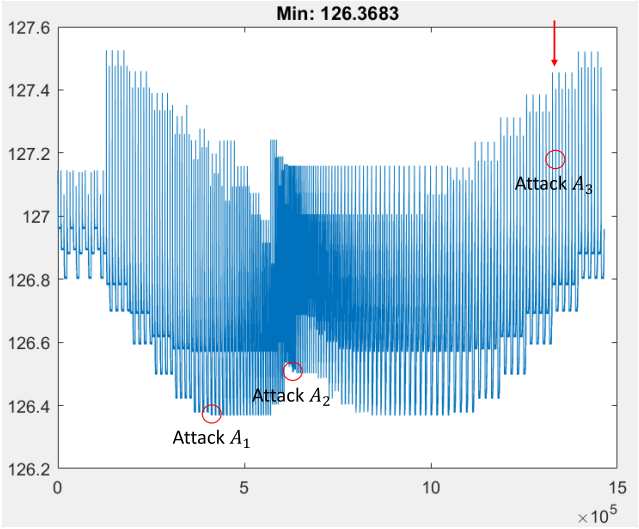


FIGURE 8. Time complexity of all generalized MITM attacks on CRAFT; vertical axis shows logarithm of the time complexity, and horizontal axis shows the attack number.

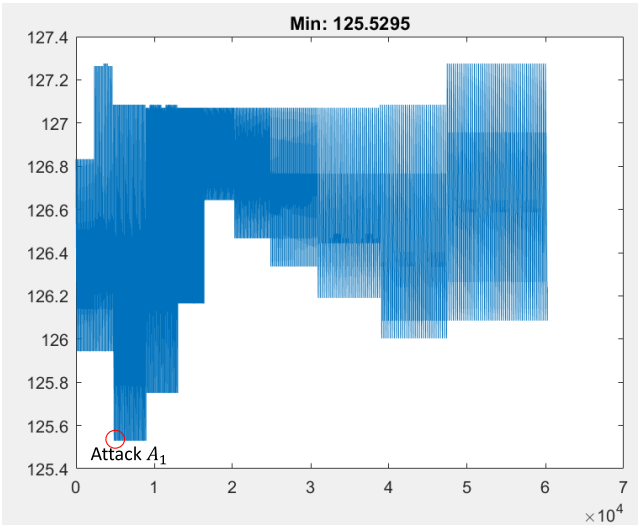


FIGURE 9. Time complexity of all generalized MITM attacks on AES-128; vertical axis shows logarithm of the time complexity, and horizontal axis shows the attack number.

- Minimum time complexity of these attacks is equal to $2^{125.3947}$ (see Figure 6), which is obtained by $\mathcal{C}_1 = (X^2_{18}, X^0_{20}, X^2_{20}, X^2_{22}, X^6_{22}, X^2_{24}, X^4_{24}, X^2_{26})$ and $\mathcal{V}_1 = X^6_{50}$.
- By $\mathcal{C}_1$ and $\mathcal{V}_1$, an attack A_1 with time, memory and data complexities of $2^{125.399}$, 2^{32} and 2^{64} can be obtained by (K^7, K^8) and (K^3, K^9, K^{10}) as forward and backward subkeys, respectively. It should be emphasized that the best known time complexity for full HIGHT was $2^{125.67}$ before [22].
- Considering cut-set $\mathcal{C}_2 = (X^1_1, X^7_1, X^0_2, X^2_2, X^6_2, X^2_4, X^4_4, X^2_6)$ and matching variable $\mathcal{V}_2 = X^4_{32}$, another attack A_2 with attack number of 294740 can be obtained by (K^2, K^3, K^{10}) and (K^4, K^7) as forward and backward subkeys, respectively. The time, memory and data complexities of the attack are equal to $2^{125.4658}$, 2^{32} and 2^{32}, respectively. The details of attack A_2 are also depicted in Figure 10. In this figure, the more thicker line, the more subkeys affected. The influence of chosen forward and backward subkeys on F-functions are shown on the top of each F-function. The light gray lines and functions are also removed in the attack procedure.

2) ENHANCEMENT OF PARTIAL MATCHING ON HIGHT ALGORITHM

By some improvements in the matching part, the attack results can be more enhanced. More precisely, the computation parts which are surrounded by dashed lines in Figure 10 can be performed in a bit-wise manner, from the least significant bit (LSB) to the most significant bit (MSB). This early abort idea makes filtering out the wrong keys faster by dropping them at the first time which a bit of matching variable is not matched from the both side. As the computation of each bit of the matching variable requires only one output bit of the all F-functions which have influenced on the dashed line parts, the complexity of computing this output bit of each F-function is only 2^{-3} F-function, referring to F-functions details in [18]. Therefore, according to early abort technique, the computational complexity of these F-functions should be $2^{-3}(1+2^{-1}+\cdots+2^{-7}) < 2^{-2}$ instead of 1 for each attack.

This technique can be applied on matching part of all generalized MITM attacks on HIGHT. The results for attack A_1 and A_2 are as follows:

- As there are 5 F-functions involved in bit-wise matching part of A_1 which activated by all subkeys, considering the bit-wise matching part results in reducing $\frac{5\times(1-2^{-2})}{128}2^{128} = 2^{122.907}$ of A_1 time complexity. Therefore, the time complexity of attack A_1 will be equal to $2^{125.399} - 2^{122.907} = 2^{125.116}$.
- It can be seen in Figure 10 that there are 7 F-functions involved in bit-wise matching part of A_2 which activated by all subkeys. Therefore, considering the bit-wise matching part results in reducing $\frac{7\times(1-2^{-2})}{128}2^{128} = 2^{123.392}$ of A_2 time complexity. Therefore, A_2 time complexity will be equal to $2^{125.4658} - 2^{123.392} = 2^{125.075}$. Surprizingly, the final time and data complexities of attack A_2 after considering bit-wised matching part are less than attack A_1. In addition, considering 4-MSB of chosen subkeys in forward and backward directions (similar to [17]) leads to lower data and memory complexities of 2^{24} and 2^{16} for attack A_2 while the new time complexity will be $2^{125.2}$.

B. PICCOLO-128

1) GENERALIZED MITM ATTACK ON PICCOLO-128

Applying generalized MITM attack on Piccolo-128 results in the followings:

- According to its simple key schedule, any random chosen cut-set in the key schedule has the same results. Therefore, without loss of generality, the master key is chosen for that cut-set.

- There are 140 points, 1461 cut-sets, and 175156 possible attacks for Piccolo-128 block cipher algorithm.
- Minimum time complexity of these attacks is equal to $2^{126.8795}$ (see Figure 7), which is obtained by $\mathcal{V}_1 = X_4^2$ and $\mathcal{C}_1 = (X_{24}^6, X_{24}^7, X_{26}^2, X_{26}^3, X_{26}^6, X_{26}^7, X_{28}^6, X_{28}^7)$.
- With the above cut-set $\mathcal{C}_1$ and matching variable $\mathcal{V}_1$, an attack A_1 with memory of 2^{24} can be obtained by $(K^{10}, K^{11}) = (k_L^5, k_R^5)$ and $(K^{12}, K^{13}) = (k_L^6, k_R^6)$ as forward and backward subkeys, respectively. The time and data complexities of attack A_1 are also equal to $2^{126.8795}$ and 2^{64}, respectively. It should be emphasized that the best known time complexity for full Piccolo-128 was $2^{127.12}$ before [17]. Considering 4-MSB of chosen subkeys in forward and backward directions and going through details of Piccolo-128 F-function computations as in [17] leads to approximately reduction of two dominant F-functions computations and new time, memory and data complexities of $2^{126.78}$, 2^{12} and 2^{64} for attack A_1, respectively.
- By choosing cut-set $\mathcal{C}_2 = (X_{56}^2, X_{56}^3, X_{58}^2, X_{58}^3, X_{58}^6, X_{58}^7, X_{60}^2, X_{60}^3)$ and matching variable $\mathcal{V}_2 = X_{12}^2$, another attack A_2 with attack number of 35302 can be obtained with (k_L^6, k_R^6) and (k_L^3, k_R^3) as forward and backward subkeys, respectively (see Figure 7). The time, memory and data complexities of the attack A_2 are equal to $2^{127.0689}$, 2^{24} and 2^{48}, respectively. Similar to attack A_1, considering 4-MSB of chosen subkeys and going through details of Piccolo-128 F-function computations leads to new time, memory and data complexities of 2^{127}, 2^{12} and 2^{40} for attack A_2, respectively.
- Trying to find attacks with data complexity of equal to 2^8 or less results in some attacks with the same complexities which one of them is exactly the one in [17]. This shows that the algorithm of "low data complexity biclique cryptanalysis of block cipher" in [17] and the algorithm of "generalized MITM attack" with the condition of low data complexity on Piccolo-128 have the same results.

C. CRAFT

1) GENERALIZED MITM ATTACK ON CRAFT

Applying generalized MITM attack on CRAFT results in the followings:

- Due to its simple key schedule, the master key is chosen for key schedule cut-set.
- There are 784 different points in CRAFT block cipher algorithm. This high number of different points results in more than millions of cut-sets which makes difficult to check all the attacks. Here by combining the successive points, which each one can be computed from any other of them assuming that the subkeys are known, in to one point, the total number of different points is reduced to 272. There are still millions of cut-sets which can be made by these 272 points. One way for reducing the number of these cut-sets can be choosing only the cut-sets which have acceptable computational complexity for exhaustive search in both direction without partial matching (just like computing computational complexity of a plaintext P to its corresponding ciphertext C, and vice versa). Applying this idea on cut-set finder (ERS) algorithm results in 347194 different cut-sets for CRAFT.
- With these 272 different points and 347194 cut-sets, there are 1464915 possible attacks for CRAFT.
- Minimum time complexity of these attacks is equal to $2^{126.3683}$ (see Figure 8), which is obtained by $\mathcal{V}_1 = X_{21}^1$ and $\mathcal{C}_1 = (X_{61}^9, X_{61}^1 3, X_{65}^0, X_{65}^{12}, X_{69}^0, X_{69}^4, X_{69}^8, X_{69}^{12}, X_{73}^1, X_{73}^4, X_{73}^5, X_{73}^8, X_{73}^9, X_{73}^{13}, X_{77}^1, X_{77}^5)$.
- With cut-set $\mathcal{C}_1$ and matching variable $\mathcal{V}_1$, an attack A_1 with time, memory and data complexities of $2^{126.3692}$, 2^{28} and 2^{64} can be obtained by $(K^6, K^{10}, K^{18}, K^{23})$ and $(K^1, K^2, K^{13}, K^{14})$ as forward and backward subkeys, respectively.
- By choosing cut-set $\mathcal{C}_2 = (X_0^3, X_0^6, X_0^{10}, X_0^{15}, X_1^4, X_1^5, X_1^8, X_1^9, X_5^8, X_5^{12}, X_9^0, X_9^1, X_9^{12}, X_9^{13}, X_{13}^9, X_{13}^{13})$ and matching variable $\mathcal{V}_2 = X_{61}^1$, another attack A_2 with attack number of 640476 can be obtained with $(K^2, K^{14}, K^{26}, K^{30})$ and $(K^6, K^{10}, K^{17}, K^{21})$ as forward and backward subkeys, respectively (see Figure 8). The time, memory and data complexities of attack A_2 are equal to $2^{126.53}$,2^{28} and 2^{48}, respectively.
- Due to the tweakable design of CRAFT, a method for 2^{-4} reduction in time complexity of exhaustive search is introduced in [20]. Combining this method with generalized MITM attack is possible. In this scenario, the computations from ciphertext to the matching point (or cut-set $\mathcal{C}_C$) and also data complexity should be increased by multiplication in 2^4. Hence, the matching point should be selected near the ciphertext as much as possible. As an example, by choosing cut-set $\mathcal{C}_3 = (X_0^2, X_0^{11}, X_0^{14}, X_1^0, X_1^{12}, X_1^{13}, X_5^0, X_5^4, X_9^4, X_9^5, X_9^8, X_9^9, X_{13}^1, X_{13}^5, X_{13}^8, X_{17}^1)$ and matching variable $\mathcal{V}_3 = X_{121}^1$, another attack A_3 with attack number of 1335048 can be obtained with $(K^6, K^{10}, K^{18}, K^{26})$ and (ϕ) as forward and backward subkeys, respectively (see Figure 8). Time, memory and data complexities of A_3 are equal to $2^{127.18}$, 2^{12} and 2^{52}, respectively. There are only one active S-box from ciphertext to matching variable $\mathcal{V}_3$. Hence, using the same method introduced in [20] results in 2^{-4} reduction plus more $\frac{1 \times (2^4 - 1)}{512} \times 2^{124} = 2^{118.91}$ in time complexity, and 2^4 times data complexity. Therefore, the final time, memory and data complexities will be $2^{127.18} \times 2^{-4} + 2^{118.91} = 2^{123.25}$, 2^{12} and $2^{52} \times 2^4 = 2^{56}$, respectively.

D. AES-128

1) GENERALIZED MITM ATTACK ON AES-128

Applying generalized MITM attack on AES-128 results in the followings:

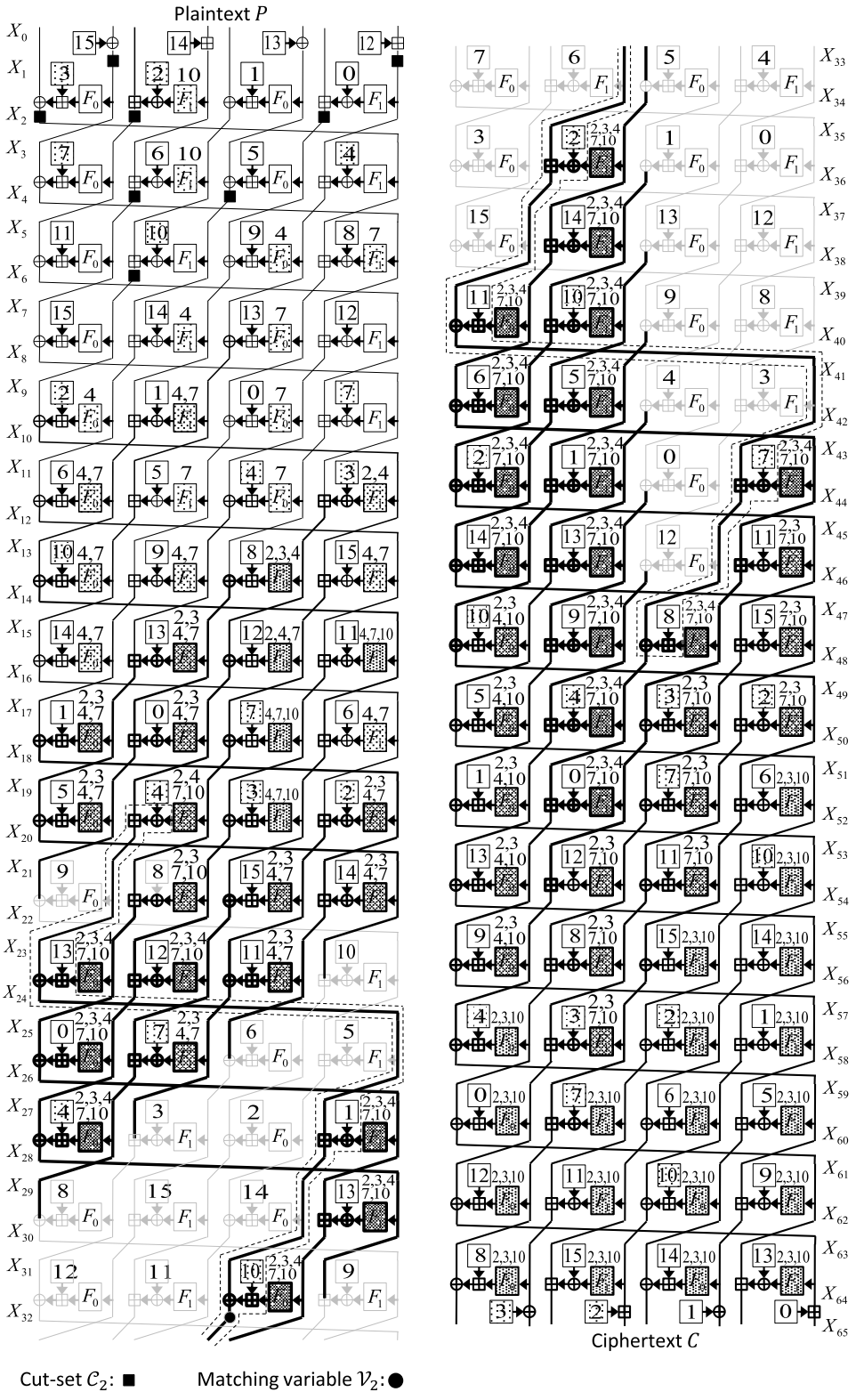


FIGURE 10. The details of generalized MITM attack on HIGHT (attack A_2).

- AES key schedule is not as simple as key schedule of previous block ciphers and any cut-set in it can be considered as the master key with different effects on subkeys. When the cut-set of sixth 128-bit subkey (K_6) is considered as the master key, there is no activated S-box by all bytes of this subkey in the key schedule.

Hence, it can be a good choice for generalized MITM attack as the computational complexity of key schedule can be ignored in this situation.

- There are 336 different points in AES-128 block cipher algorithm. However, there are millions of cut-sets which can be made by these points. Therefore, a similar method as CRAFT is used for sieving these cut-sets. Applying this idea on cut-set finder algorithm results in 267 different cut-sets and 60256 possible attacks for AES-128 block cipher algorithm.
- Minimum time complexity of these attacks is equal to $2^{125.5295}$ (see Figure 9), which is obtained by $\mathcal{V}_1 = X_4^0$ and $\mathcal{C}_1 = (X_{20}^0, X_{20}^4, X_{20}^{12}, X_{21}^1, X_{21}^3, X_{21}^5, X_{21}^6, X_{21}^9, X_{21}^{10}, X_{21}^{11}, X_{21}^{14}, X_{21}^{15}, X_{24}^8, X_{25}^9, X_{25}^{10}, X_{25}^{11})$. This complexity is less than what has been found before in [23].
- Using $\mathcal{C}_1$ and $\mathcal{V}_1$, an attack A_1 with time, memory and data complexities of $2^{125.5696}$, 2^{32} and 2^{128} can be obtained by $(K_6^9, K_6^{10}, K_6^{11})$ and (K_6^0, K_6^{12}) as forward and backward subkeys, respectively.

VI. CONCLUSION

In this paper, the generalized MITM attack is proposed and automated by searching around cut-sets of a block cipher. Then, it is applied on four different block cipher, HIGHT and Piccolo-128 with Feistel structure, and CRAFT and AES with SPN structure. The results show some improvements in comparison to previous works. As an example, generalized MITM attack along with bit-wised matching part on HIGHT has found an attack with time, memory and data complexities of $2^{125.2}$, 2^{16} and 2^{24}, which has less time and data complexities than the previous work on full-round HIGHT [22], with time, memory and data complexities of $2^{125.67}$, 2^{16} and 2^{42}, respectively. In addition, generalized MITM attack on Piccolo-128 has found the same result as [17] when data complexity limitation is applied on the attack. Also, generalized MITM attack on AES-128 approved possibility of finding out an attack with less time complexity than the attack mentioned in [23].

Generalized MITM attack on block ciphers can be performed along with some other ideas such as early abort, sieve in the middle, or innovative techniques to improve the attack results. As an example, in the attack on full HIGHT, considering the idea of bit-wised matching part along with 4-MSB bits of subkeys leads to a new attack with interesting results in which all the complexities of the attack are lower than or equal to the best previous known attack. Another example is in the attack on CRAFT which is a tweakable block cipher. Generalized MITM attack on CRAFT along with improved exhaustive search introduced in [20] leads to a new attack with lower time complexity of $2^{123.25}$ in comparsion to 2^{124} of designer's analysis. It should be highlighted that the genealized MITM attack can cover almost all previous versions of MITM attack including biclique attack. Therefore, providing security against generalized MITM attack can confirm the security against these previous versions. Another advantage of generalized MITM attack is automatic key partitioning during the attack procedure.

ACKNOWLEDGMENT

The authors would like to thank Information Security and Systems Lab (ISSL) members, specially J. Alizade, considerable and valuable comments.

REFERENCES

[1] W. Diffie, and M. E. Hellman, "Special feature exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, vol. 10, no. 6, pp. 74–84, 1977.
[2] A. Bogdanov and C. Rechberger, "A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN," in *Selected Areas in Cryptography* (Lecture Notes in Computer Science), vol. 6544, A. Biryukov, G. Gong, and D. Stinson, Eds. Berlin, Germany: Springer, 2010, pp. 229–240.
[3] L. Wei, C. Rechberger, J. Guo, H. Wu, H. Wang, and S. Ling, "Improved meet-in-the-middle cryptanalysis of KTANTAN (poster)," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany: Springer, 2011, pp. 433–438.
[4] A. Bogdanov, D. Khovratovich, C. Rechberger, *Biclique Cryptanalysis of the Full AES*, (Lecture Notes in Computer Science), vol. 7073. Heidelberg, Germany: Springer, 2011, pp. 344–371.
[5] G. Han, H. Zhao, and C. Zhao, "Unbalanced Biclique cryptanalysis of full-round GIFT," *IEEE Access*, vol. 7, pp. 144425–144432, 2019, doi: 10.1109/ACCESS.2019.2945006.
[6] J. Lu, J. Kim, N. Keller, and O. Dunkelman, "Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1," in *Proc. Cryptographers Track RSA Conf.*, San Francisco, CA, USA, Apr. 2008.
[7] J. Lu, J. Kim, N. Keller, and O. Dunkelman "Sieve-in-the-middle: Improved MITM attacks," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 8042, R. Canetti and J. Garay, Eds. Berlin, Germany: Springer, 2008, pp. 222–240.
[8] K. Fu, M. Wang, Y. Guo, S. Sun, and L. Hu, "MILP-based automatic search algorithms for differential and linear trails for speck," in *Proc. Int. Conf. Fast Softw. Encryption*, vol. 21, 2016, p. 27.
[9] S. Siwei, "Towards finding the best characteristics of some bit-oriented block ciphers and automatic enumeration of (related-key) differential and linear characteristics with predefined properties," Cryptol. ePrint Arch., Tech. Rep. 2014/747, 2014.
[10] J. Chen, J. Teh, Z. Liu, C. Su, A. Samsudin, and Y. Xiang, "Towards accurate statistical analysis of security margins: New searching strategies for differential attacks," *IEEE Trans. Comput.*, vol. 66, no. 10, pp. 1763–1777, Oct. 2017.
[11] L. Song, Z. Huang, and Q. Yang, "Automatic differential analysis of ARX block ciphers with application to SPECK and LEA," in *Information Security and Privacy* (Lecture Notes in Computer Science), vol. 9723, J. K. Liu and R. Steinfeld, Eds. Cham, Switzerland: Springer, 2016, pp. 379–394.
[12] S. Wu and M. Wang, "Automatic search of truncated impossible differentials for word-oriented block ciphers," in *Proc. Int. Conf. Cryptol. India*, 2012, pp. 283–302.
[13] D. Patrick, and P.-A. Fouque, "Automatic search of meet-in-the middle and impossible differential attacks," in *Proc. Annu. Int. Cryptol. Conf.*, Berlin, Germany: Springer, 2016.
[14] L. Lin, W. Wu, and Y. Zheng, "Automatic search for key-bridging technique: Applications to LBlock and TWINE," in *Proc. Int. Conf. Fast Softw. Encryption*, 2016, pp. 247–267.
[15] A. Farzaneh, "A framework for automated independent-biclique cryptanalysis," in *Proc. Int. Workshop Fast Softw. Encryption*, Berlin, Germany: Springer, 2013.
[16] H. Kai, and M. Wang, "Automatic search for a variant of division property using three subsets," in *Proc. Cryptographers Track RSA Conf.*, Cham, Switzerland, Springer, 2019.
[17] S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref. "Low-data complexity biclique cryptanalysis of block ciphers with application to piccolo and hight," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 10, pp. 1641–1652, Oct. 2014.

[18] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee, *HIGHT: A New Block Cipher Suitable For low-Resource Device. Cryptographic Hardware and Embedded Systems-CHES. Lecture Notes in Computer Science*, vol. 4249, Berlin, Germany: Springer, pp. 46–59, 2006.

[19] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T Shirai, *Piccolo: An Ultra-Lightweight Blockcipher* (Lecture Notes in Computer Science), vol. 6917. Heidelberg, Germany: Springer, 2011, pp. 342–357.

[20] B. Christof, "CRAFT: Lightweight tweakable block cipher with efficient protection against DFA Attacks," *IACR Trans. Symmetric Cryptol.*, vol. 2019, no. 1, pp. 5–45, 2019.

[21] J. Daemen and V. Rijmen, *The Design of Rijndael: AES—The Advanced Encryption Standard* (Information Security and Cryptography). Berlin, Germany: Springer, 2002.

[22] S. A. Azimi, S. Ahmadi, Z. Ahmadian, J. Mohajeri, and M. R. Aref, "Improved impossible differential and biclique cryptanalysis of hight," *Int. J. Commun. Syst.*, vol. 31, no. 1, p. e3382, 2018.

[23] B. Andrey, "Bicliques with minimal data and time complexity for AES," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Cham, Switzerland: Springer, 2014.

[24] B. Tao and H. Wu, "Improving the biclique cryptanalysis of AES," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Cham, Switzerland: Springer, 2015.

[25] A. E. Moghaddam and Z. Ahmadian, "New automatic search method for truncated-differential characteristics: Application to Midori and SKINNY," IACR Cryptol. ePrint Archive Tech. Rep., 2019, p. 126.

[26] H. Hadipour, S. Sadeghi, M. M. Niknam, and N. Bagheri. "Comprehensive security analysis of CRAFT," Cryptology ePrint Archive, Tech. Rep. 2019/741, 2019.

[27] M. ElSheikh, and A. M. Youssef, "Related-key differential cryptanalysis of full round CRAFT," IACR Cryptol. ePrint Archive Tech. Rep., 2019, p. 932.

[28] C. Bouillaguet, P. Derbez, and P.-A. Fouque, "Automatic search of attacks on round-reduced AES and applications," in *Proc. Annu. Cryptol. Conf.*, Berlin, Germany: Springer, 2011.

SIAVASH AHMADI received the B.S. and M.S. degrees in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2012 and 2014, respectively, where he is currently pursuing the Ph.D. degree in electrical engineering (communication systems and security). His special fields of interest include cryptology and wireless security, with emphasis on cryptanalysis.

MOHAMMAD REZA AREF received the B.S. degree from the University of Tehran, Iran, in 1975, and the M.Sc. and Ph.D. degrees from Stanford University, Stanford, CA, USA, in 1976 and 1980, respectively, all in electrical engineering. He returned to Iran, in 1980 and was actively engaged in academic affairs. He was a Faculty Member with the Isfahan University of Technology, from 1982 to 1995. He has been a Professor of electrical engineering with the Sharif University of Technology, Tehran, since 1995. He has published more than 290 technical articles in communication and information theory and cryptography in international journals and conferences proceedings. His current research interests include areas of communication theory, information theory, and cryptography.