Automatic Search for Key-Bridging Technique: Applications to LBlock and TWINE

Li Lin[1,2,3], Wenling Wu[1,2,3](✉), and Yafei Zheng[1,2,3]

[1] TCA Laboratory, SKLCS, Institute of Software, Chinese Academy of Sciences,
Beijing, China
{linli,wwl,zhengyafei}@tca.iscas.ac.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] Graduate University of Chinese Academy of Sciences, Beijing 100190, China

Abstract. Key schedules in block ciphers are often highly simplified, which causes weakness that can be exploited in many attacks. At ASIACRYPT 2011, Dunkelman et al. proposed a technique using the weakness in the key schedule of AES, called key-bridging technique, to improve the overall complexity. The advantage of key-bridging technique is that it allows the adversary to deduce some sub-key bits from some other sub-key bits, even though they are separated by many key mixing steps. Although the relations of successive rounds may be easy to see, the relations of two rounds separated by some mixing steps are very hard to find. In this paper, we describe a versatile and powerful algorithm for searching key-bridging technique on word-oriented and bit-oriented block ciphers. To demonstrate the usefulness of our approach, we apply our tool to the impossible differential and multidimensional zero correlation linear attacks on 23-round LBlock, 23-round TWINE-80 and 25-round TWINE-128. To the best of our knowledge, these results are the currently best results on LBlock and TWINE in the single-key setting.

Keywords: Block ciphers · Key-bridging technique · Automatic search · Impossible differential cryptanalysis · Zero-correlation linear cryptanalysis · LBlock · TWINE

1 Introduction

A key schedule is an algorithm that expands a relatively short master key to a relatively large expanded key for later use in encryption and decryption algorithms. The key schedules in block ciphers are often highly simplified, which causes weakness that can be exploited in many attacks, especially for lightweight block ciphers. In these lightweight block ciphers, the security margin that conventional block ciphers are equipped with is reduced as much as possible in order to optimize the software and hardware efficiency. One obvious sacrifice is that the key schedules are highly simplified for saving memory. Some key schedules have round-by-round iterations with low diffusion [4,21,25]. Some key schedules do simple permutations or linear operations with low diffusion [1,13]. Some have no

T. Peyrin (Ed.): FSE 2016, LNCS 9783, pp. 247–267, 2016.
DOI: 10.1007/978-3-662-52993-5_13

key schedules, and just use master keys directly in each round [12,15]. These key schedules are succinct but responsible for many attacks, especially related-key attacks [3,16] and meet-in-the-middle attacks [2,5].

AES [8] is the most significant standard for block ciphers, so its security is of paramount importance. However, the key schedule of AES has clear weakness that directly assists the execution of some effective attacks. Especially in recent years, meet-in-the-middle cryptanalysis with differential enumeration technique [11] has shown to be a very powerful form of cryptanalysis against 7-round AES-128 [9], 9-round AES-192 [17] and 10-round AES-256 [18], which are the best single-key attacks on all versions of AES so far. A technique using the weakness of the key schedule on AES, called key-bridging technique, is used in these attacks to improve the overall complexity. Key-bridging technique is proposed by Dunkelman et al. at ASIACRYPT 2011 [11]. The advantage of key-bridging technique is that it allows the adversary to deduce some sub-key bytes from some other sub-key bytes, even though they are separated by many key mixing steps. Although the relations of successive rounds may be easy to see, the relations of two rounds separated by some mixing steps are very hard to find. The main novelty in this observation is that it exploits the weak key schedule of AES-192 in order to provide a surprisingly long "bridge" for two sub-keys which are separated by 8 key mixing steps. The key-bridging technique considerations reduce the time complexity in the online phase of the attack on 8-round AES-192 by a factor of 2^{32} and 8-round AES-256 by a factor of 2^{8}. At FSE 2014, Li et al. introduce a new application of key-bridging technique called key-dependent sieve technique, which filters the wrong states based on the key relations, to further reduce the complexity in the precomputation phase [17]. Besides, they introduce another application of key-bridging technique to split the whole attack into some weak-key attacks according to the relations between the sub-keys in the online phase and the precomputation phase.

Besides AES, the key-bridging technique helps improve the attack complexities of other block ciphers. For example, at FSE 2015, Biryukov et al. apply the key-bridging technique to 25-round TWINE-128, and get a meet-in-the-middle attack and an impossible differential attack [2]. At ACISP 2014, Wang et al. give multidimensional zero-correlation linear attacks on LBlock and TWINE. In the online phase of their attacks, the key-bridging technique is used to improve the attack complexity [22].

Our Contribution. In this paper, we describe versatile and powerful algorithms for searching key-bridging technique on word-oriented and bit-oriented block ciphers. Our tool tries to find key-bridges automatically by dealing with a system of equations. It takes as input a system of equations that describes the key schedule and a set $\mathbb{K}_0$ of some key variables that we want to build key-bridges among. It is made up of two phases: knowledge-propagation phase and relation-derivation phase. In the knowledge-propagation phase, we can derive a set $\mathbb{K}$ that $\mathbb{K}_0$ can propagate to. In the relation-derivation phase, the relations of the variables in $\mathbb{K}_0$ can be known from $\mathbb{K}$.

To demonstrate the usefulness of our approach, we apply our tool to LBlock and TWINE. We automatize the search for the best impossible differential attacks by combining our key-bridging tool with the tool of Wu et al. [24]. Using Wu's tool, we can get all the impossible differential distinguishers with certain rounds [20]. Using our key-bridging tool, we can get all the key-bridges to reduce the complexity in the key-sieving phase. With these two tools, we get a 23-round impossible differential attack on LBlock with time complexity of $2^{74.5}$ 23-round LBlock encryptions, memory complexity of $2^{74.3}$ bytes and data complexity of $2^{59.5}$ chosen plaintexts. For TWINE-128, we get in total twelve 25-round impossible differential attacks with the same complexity as Biryukov et al.'s attack in [2].

For multidimensional zero-correlation linear cryptanalysis, we use the same attack model Wang et al. proposed in [22] and get more key-bridges to improve the overall complexity with our key-bridging tool. For the 23-round attack on TWINE-80, we find that the key-bridges Wang et al. used in their attack do not exist. This will make the time complexity of their attack greater than exhaustively search. We use another zero-correlation linear distinguisher to fix this error and get an attack on 23-round TWINE-80 with time complexity of 2^{73} 23-round TWINE-80 encryptions, memory complexity of 2^{60} bytes and data complexity of $2^{62.1}$ known plaintexts. For the 25-round attack on TWINE-128, we first get some more key-bridges to improve the time complexity of Wang's work. Then, we use another distinguisher with more key-bridges in the first two steps of the attack, and get an attack with time complexity of 2^{119} 25-round TWINE-128 encryptions, memory complexity of 2^{60} bytes and data complexity of $2^{62.1}$ known plaintexts. For the 23-round multidimensional zero-correlation linear attack on LBlock, we find a distinguisher with more key-bridges than Wang et al. in [24], and get an attack with time complexity of 2^{72} 23-round LBlock encryptions, memory complexity of 2^{60} bytes and data complexity of $2^{62.1}$ known plaintexts. For 25-round TWINE-128, we also find some meet-in-the-middle attacks with the same complexity as Biryukov et al.'s attack in [2], even with one attack which starts with two inactive nibbles at the beginning of distinguisher. This distinguisher is useful when we want to get less false positive. To the best of our knowledge, these results are the currently best results on LBlock and TWINE.

We present here a summary of our attack results on LBlock and TWINE, and compare them to the best attacks known for them. This summary is given in Table 1. The source code of some of these attacks is available at https://onedrive.live.com/redir?resid=20C3554F0C8B0806!108&authkey=!AJPOWJTJ4mSLrbI&ithint=folder%2c7z.

Organization of This Paper. The rest of this paper is organized as follows. Section 2 presents the input of our tool and the previous works on key-bridging technique. Section 3 gives our automatic search tool for key-bridging technique. Section 4 (resp. Sect. 5) applies our tool to the impossible differential and multidimensional zero-correlation linear cryptanalysis on LBlock (resp. TWINE). Finally, Sect. 6 concludes this paper.

Table 1. Summary of the best attacks on LBlock and TWINE-80/128.

Cipher	Attack type	Rounds	Data	Memory (Bytes)	Time (Enc)	Source
LBlock	Impossible Diff.	23	2^{59} CPs	2^{74}	$2^{75.36}$	[6]
	Impossible Diff	23	$2^{59.5}$ CPs	$2^{74.3}$	$2^{74.5}$	Sect. 4.2
	Multidim. ZC	23	$2^{62.1}$ KPs	2^{60}	2^{76}	[22]
	Multidim. ZC	23	$2^{62.1}$ KPs	2^{60}	2^{72}	Sect. 4.3
TWINE-80	Impossible Diff	23	$2^{57.85}$ CPs	$2^{84.06}$	$2^{79.09}$	[26]
	Multidim. ZC	23	$2^{62.1}$ KPs	2^{60}	2^{73}	Sect. 5.2
TWINE-128	Impossible Diff.*	25	$2^{59.1}$ CPs	$2^{78.1}$	$2^{124.5}$	[2]
	MITM *	25	2^{48} CPs	2^{109}	$2^{124.7}$	[2]
	Multidim. ZC	25	$2^{62.1}$ KPs	2^{60}	$2^{122.12}$	[22]
	Multidim. ZC	25	$2^{62.1}$ KPs	2^{60}	2^{119}	Sect. 5.2

KPs: Known-Plaintexts. CPs: Chosen-Plaintexts.
*: Find the other attacks with the same complexity in Sect. 5.3.

2 Preliminaries

In this section, we introduce the definitions and related works of key-bridging technique. First of all, let's give some notations used throughout this paper.

2.1 Notations

In this paper, WK_i denotes the i^{th} round key register, WK_i^j denotes the j^{th} word of WK_i, $WK_i^{j_0-j_1}$ denotes the j_0^{th} word to j_1^{th} word of WK_i, $WK_i[k]$ denotes the k^{th} bit of WK_i, $WK_i^j[k]$ denotes the k^{th} bit of WK_i^j and $WK_i \lll b$ denotes b-bit left cyclic shift of WK_i.

2.2 The Key Schedule Functions

The input of our tool is a system of equations that describes the key schedule and the key variables which we want to find relations among. Since our tool is useful not only for the word-oriented key schedules (e.g., AES), but also for the bit-oriented key schedules (e.g., PRESENT), we describe the systems of equations for these two kinds of key schedules here. We take the key schedules of AES-192 and PRESENT-80 as examples.

The key schedule of AES-192 takes the 192-bit master key WK_0 and extends it into 9 key registers $WK_0, WK_1, \cdots, WK_8$ of 192-bit each using a key schedule algorithm given by the following equations [8]:

$$KS_i : \begin{cases} WK_i^j + WK_i^{j-4} + WK_{i-1}^j = 0, j = 4, \cdots, 23, \\ WK_i^0 + WK_{i-1}^0 + S(WK_{i-1}^{21}) + RCON_i = 0, \\ WK_i^1 + WK_{i-1}^1 + S(WK_{i-1}^{22}) = 0, \\ WK_i^2 + WK_{i-1}^2 + S(WK_{i-1}^{23}) = 0, \\ WK_i^3 + WK_{i-1}^3 + S(WK_{i-1}^{20}) = 0, \end{cases}$$

where S represents the S-box of the `SubBytes` transformation and "+" represents xor. These 9 key registers are used to get 13 sub-keys $RK_{-1}, RK_0, \cdots, RK_{11}$ of 128-bit each (only the first 128-bit of WK_8 is used to get RK_{11}).

In some cases, we are interested in interchanging the order of the `MixColumns` and `AddRoundKey` operations. As these operations are linear, they can be interchanged by first xoring the data with an equivalent key u_i and then applying the `MixColumns` operation.

The key schedule of PRESENT-80 takes the 80-bit master key WK_0 and extends it into 32 key registers $WK_0, WK_1, \cdots, WK_{31}$ of 80 bits each using a key schedule algorithm given by the following equations [4]:

$$KS_i : \begin{cases} WK_i[0-3] + S(WK_{i-1}[61-64]) = 0, \\ WK_i[60-64] + WK_{i-1}[41-45] + [i-1] = 0, \\ WK_i[j] + WK_{i-1}[(j+19) mod\ 80] = 0, \quad j = 4, \cdots, 40, 46, \cdots, 79 \end{cases}$$

At round i, the 64-bit round key $RK_i = RK_i[0]RK_i[1] \cdots RK_i[63]$ consists of the 64 leftmost bits of the current content of register WK_i.

The key schedules of other bit-oriented and word-oriented block ciphers can be treated as before. To simplify the statement, we ignore the round constants in this paper since they are known to us.

2.3 Key-Bridging Technique on AES

In [11], Dunkelman et al. proposed the key bridging technique on AES-192. The advantage of key-bridging technique is that it allows the adversary to deduce some sub-key bytes from some other sub-key bytes, even though they are separated by many key mixing steps. Although the relations of successive rounds may be easy to see, the relations of two rounds separated by some mixing steps are very hard to find. The main novelty in this observation is that it exploits the weak key schedule of AES-192 in order to provide a surprisingly long "bridge" for two sub-keys which are separated by 8 key mixing steps (applied in reverse direction). This observation is shown in Observation 1.

Observation 1 (Key-Bridging Technique on AES, [11]). *By the key schedule of AES-192, knowledge of columns 0, 1, 3 of the sub-key RK_7 allows to deduce column 3 of the whitening key RK_{-1} (which is actually column 3 of the master key).*

Given RK_7^{0-3} and RK_7^{4-7}, it is possible to compute RK_5^{12-15}; given RK_7^{4-7} and RK_7^{12-15}, it is possible to compute RK_4^{12-15}. From these two values, it is possible to compute RK_{-1}^{12-15}. We refer to [10] the detailed proof and reasoning.

The key-bridging technique considerations reduce the time complexity of the online phase of the attacks on 8-round AES-192 by a factor of 2^{32} and 8-round AES-256 by a factor of 2^8 [10], and also improve the SQUARE attack and related-key impossible differential attack on AES-192.

At EUROCRYPT 2013, Derbez et al. gave improved attacks on 7-round AES-128, 8-round AES-192 and 9-round AES-256 [9]. In the online phase of their attack on 8-round AES-192, the use of the key-bridging technique saves a large amount of time.

At FSE 2014, Li et al. introduced a new application of key-bridging technique called **key-dependent sieve technique**, which filters the wrong states based on the key relations, to further reduce the complexity in the precomputation phase [17]. Besides, they found that the whole attack can be split up into some weak-key attacks according to the relations between the sub-keys in the online phase and the precomputation phase. These can be seen as other applications of key-bridging technique.

In [18], Li and Jen gave an attack on 10-round AES-256. In their works, they use key-bridging technique both in the precomputation phase and the online phase.

2.4 Key-Bridging Technique on Other Block Ciphers

At FSE 2015, Biryukov et al. applied the key-bridging technique to 25-round TWINE-128, and got a meet-in-the-middle attack and an impossible differential attack [2].

In the meet-in-the-middle attack, 58 state nibbles are needed to perform the online phase. Hopefully, the key schedule equations reduce the amount of possible values from $2^{4\times 58} = 2^{232}$ to 2^{124}. Indeed, knowing 23 out of 24 nibbles of one sub-key leads to the knowledge of enough key material to partially encrypt and decrypt the plaintext and the ciphertext in order to obtain the value of the required state variables. This can be seen as 37 key-bridges among the 68 relevant sub-key nibbles. The same technique is applied to the impossible differential attack.

At ACISP 2014, Wang et al. gave multidimensional zero-correlation linear attacks on LBlock and TWINE. In the online phase of their attacks, the key-bridging technique is used to reduce the overall complexity [22].

Most attacks on block ciphers can be split into three consecutive parts of r_1, r_2 and r_3 rounds, $r = r_1 + r_2 + r_3$, such that a particular set of messages may verify a certain property in the middle r_2 rounds by guessing some key-bits in the first r_1 and last r_3 rounds. These key-bits may have some relations by the key schedule. If we can get these relations automatically, it can not only give better attacked-rounds and complexity, but also a better understanding of the design of block ciphers. Therefore, we give our automatic search tool for key-bridging technique in the next section.

3 An Automatic Search Tool for Key-Bridging Technique

In this section, we introduce our automatic search tool for key-bridging technique on word-oriented and bit-oriented block ciphers.

3.1 Outline of the Tool

Let us denote by $\mathcal{V}(X)$ the vector space spanned by $1, x, S(x)$ for all $x \in X$, for any set of variables X. If we denote by $\mathbb{X}$ the set of all internal key variables, then the key schedule equations can be seen as a subspace of $\mathcal{V}(\mathbb{X})$. We introduce the notation $\mathbb{K}_0$ to denote the set of original variables that we want to build key-bridges among. We also introduce $\mathbb{K}$ to denote the set of variables that $\mathbb{K}_0$ can propagate to. And $|X|$ means the number of variables in a set X.

Our goal is to find relations among a set of variables. The difficulty in finding such relations is how to get more information from $\mathbb{K}_0$ and how to use this information to retrieve the relations. In this section, we present a tool that finds such attacks automatically. It takes as input a system of equations $\mathbb{E} \subseteq \mathcal{V}(\mathbb{X})$ that describes the key schedule and a set of variables $\mathbb{K}_0$ that we want to find relations among. This tool consists of two phases: knowledge-propagation phase and relation-derivation phase. In the knowledge-propagation phase, we can derive a set $\mathbb{K}$ that $\mathbb{K}_0$ can propagate to. In the relation-derivation phase, the relations of the variables in $\mathbb{K}_0$ can be known from $\mathbb{K}$.

In the knowledge-propagation phase, if we substitute the values of $\mathbb{K}$ into the original equations $\mathbb{E}$, we would indeed get a system of equations with less variables. In fact, this reduced system is the subspace $(\mathbb{E} + \mathcal{V}(\mathbb{K}))/\mathcal{V}(\mathbb{K})$ of the quotient space $\mathcal{V}(\mathbb{X})/\mathcal{V}(\mathbb{K})$: starting from an equation $f \in \mathbb{E}$, its equivalence class $[f]$ in the quotient contains a representative where all the variables in $\mathbb{K}$ have disappeared. Let's denote by $\mathscr{L}$ a linear combination of some variables in $\mathcal{V}(\mathbb{K})$. The variable x can be deduced from $\mathbb{K}$ if there exists an $\mathscr{L}$ such that $x + \mathscr{L} \in \mathbb{E}$, $S(x) + \mathscr{L} \in \mathbb{E}$ or the linear combination of x, $S(x)$ and $\mathscr{L}$ belongs to $\mathbb{E}$, and we will write $x \in \textsc{Propagate}(\mathbb{K})$ when it is the case. It follows that in any solution of the equations $\mathbb{E}$, the value of x (resp. $S(x)$) is the value of $\mathscr{L}$. Therefore, it just has to evaluate $\mathscr{L}$ to uniquely determine the value of x.

In the relation-derivation phase, the subspace $\mathbb{E} \cap \mathcal{V}(\mathbb{K})$ of $\mathcal{V}(\mathbb{K})$ should be derived. Then the linear relations among $\mathbb{K}_0$ can be known by dealing with the quotient space $(\mathbb{E} \cap \mathcal{V}(\mathbb{K}))/\mathcal{V}(\mathbb{K}_0)$.

3.2 A Tool for Word-Oriented Ciphers

Knowledge-Propagation Phase.[1] Let's denote by $\mathbb{M}$ the coefficient matrix made by the key schedule equations $\mathbb{E}$. Each row is a function, and each column is a variable. The order of variables is $(\mathbb{X} - \mathbb{K}, \mathbb{K}, c)$, where $\mathbb{X} - \mathbb{K}$ means the supplementary set of $\mathbb{K}$ in $\mathbb{X}$. We ignore the constant column in the matrix to better describe and express our tool in the rest of this paper. We can also view the constant column as a special column vector which always exists in the last of the matrix.

Given $\mathbb{K}_0$, we may propagate knowledge and derive the values of new variables, and this yields a new set $\mathbb{K}$. But it may turn out that new variables may again be derived from $\mathbb{K}$. The problem boils down to getting new variables and using these variables to get more information.

[1] A similar idea of this phase was proposed in [14] by Khovratovich et al..

Gauss-Jordan Elimination (GJE) is introduced to propagate knowledge [23,27]. GJE is an algorithm for solving systems of linear equations. It is usually understood as a sequence of elementary row operations performed on the associated matrix of coefficients. This method can also be used to find the rank of a matrix, and to convert a matrix into reduced row echelon form. $GJE(\mathbb{M})$ means that we convert a matrix $\mathbb{M}$ into reduced row echelon form by Gauss-Jordan Elimination. $GJE_n(\mathbb{M})$ means that we only convert the first n columns into reduced row echelon form by the row operations of the whole $\mathbb{M}$.

Since the equations $\mathbb{E}$ can be completely linear (e.g., key schedule of Simon) or partial-nonlinear (e.g., key schedule of AES), some variables appear both linearly and under the S-box. The following three situations can be used to propagate knowledge:

1. If either x or $S(x)$ belongs to $\mathbb{K}$, then the other one can be deduced.
2. If there exists a linear combination $\mathscr{L}$ of $\mathcal{V}(\mathbb{K})$ such that for one variable $x \notin \mathbb{K}$, $x + \mathscr{L} \in \mathbb{E}$, then x can be deduced from $\mathbb{K}$.
3. If there is a linear combination $\mathscr{L}$ of $\mathcal{V}(\mathbb{K})$ such that there is a linear combination of x ($x \notin \mathbb{K}$), $S(x)$ ($S(x) \notin \mathbb{K}$) and $\mathscr{L}$ belonging to $\mathbb{E}$, then x can be deduced from $\mathbb{K}$.

Gauss-Jordan Elimination is used to deal with situation 2 and situation 3 as follows. The proof of the following two lemmas will be presented in the full version of this paper.

Lemma 1. *Situation 2 holds if and only if there is only one non-zero variable in the first $|\mathbb{X} - \mathbb{K}|$ columns of one row in $GJE_{|\mathbb{X}-\mathbb{K}|}(\mathbb{M})$.*

Lemma 2. *Situation 3 holds if and only if one of the following two cases holds in $GJE_{|\mathbb{X}-\mathbb{K}|}(\mathbb{M})$ (for x and $S(x)$):*

(i) The coefficients of x and $S(x)$ are both pivot elements, and the corresponding rows are $(\underbrace{0,\cdots,0}_{t_1}, e_x, e_{t_1+1}, \cdots, e_{n-1})$ and $(\underbrace{0,\cdots,0}_{t_2}, e'_{S(x)}, e'_{t_2+1}, \cdots, e'_{n-1})$ ($t_1 < t_2$), where $e_{t_1+1} = e'_{t_1+1} = \cdots = e_{t_2-1} = e'_{t_2-1} = 0$, $e_i = c \cdot e'_i$ for $i = t_2+1, \cdots, n-|\mathbb{K}|-1$.

(ii) One of the coefficients of x and $S(x)$ is pivot element (e.g., x) and the corresponding row is $(\underbrace{0,\cdots,0}_{t_1}, e_x, \underbrace{0,\cdots,0}_{t_2}, e_{S(x)}, \underbrace{0,\cdots,0}_{n-2-t_1-t_2-|\mathbb{K}|}, e_{n-|\mathbb{K}|}, \cdots, e_{n-1})$.

When a new variable x (resp. $S(x)$) becomes a member of $\mathbb{K}$, we have to move the column that x (resp. $S(x)$) represents in $\mathbb{M}$ to the last few columns to make the order of variables $(\mathbb{X}-\mathbb{K}, \mathbb{K}, c)$ unchanged. If one of these coefficients is pivot element, then moving it may leave the matrix not in reduced row echelon form. This can be fixed through simple column permutations in some cases. In some other cases, a new column has to be recomputed. The following lemma will make sure that the column operations don't change the property of linear relations.

Lemma 3. *Column operations keep the linear relations we get from situation 2 and 3 unchanged, i.e., these relations can be recovered from the last* $|\mathbb{K}|+1$ *columns of* $GJE_{|\mathbb{X}-\mathbb{K}|}(\mathbb{M})$*.*

Proof. If situation 2 holds, then there is one new $x \in$ Propagate$(\mathbb{K})$. After moving the corresponding column and getting a new $\mathbb{K}$, there exists a vector α such that $\mathbb{K}\cdot\alpha^T = c$ and the component of α for x is non-zero (here we treat $\mathbb{K}$ as a vector and α is not unique).

After $GJE_{|\mathbb{X}-\mathbb{K}|}$, the matrix can be represented as block matrix $\begin{pmatrix} \overbrace{A_0}^{|\mathbb{X}-\mathbb{K}|} & \overbrace{A_1}^{|\mathbb{K}|} \\ 0 & A_2 \end{pmatrix}$.

Since $\mathbb{K}\cdot\alpha^T$ has no variables in $\mathbb{X}-\mathbb{K}$, the row represents this equation must exist in A_2. If not, there must be a pivot element in $\mathbb{X}-\mathbb{K}$.

Situation 3 can be got for the same reason.

□

The pseudo-code of the knowledge-propagation phase is shown in Algorithm 1. The inputs are a set of all internal key variables, a set of original variables that we want to build key-bridges among and a coefficient matrix made by the key schedule equations. The algorithm returns a set of variables that $\mathbb{K}_0$ can propagate to and a block matrix $\begin{pmatrix} \overbrace{A_0}^{|\mathbb{X}-\mathbb{K}|} & \overbrace{A_1}^{|\mathbb{K}|} \\ 0 & A_2 \end{pmatrix}$ in order to recover all the relations in $\mathbb{K}_0$.

From Lemmas 1 and 2, we can conclude that $\mathbb{K}$ is the maximum set $\mathbb{K}_0$ can propagate to. From Lemma 3, all the relations can be recovered from $\mathbb{M}$.

Relation-Derivation Phase. The input of this phase is the output of Algorithm 1. First of all, we should derive the linear relations among $\mathbb{K}$. Since the output matrix $\mathbb{M}$ of Algorithm 1 has the form $\begin{pmatrix} \overbrace{A_0}^{|\mathbb{X}-\mathbb{K}|} & \overbrace{A_1}^{|\mathbb{K}|} \\ 0 & A_2 \end{pmatrix}$, due to the proof of Lemma 3, all the linear relations among $|\mathbb{K}|$ exist in A_2. Meanwhile, $Rank(\mathbb{M}) = Rank(A_0) + Rank(A_2)$. We should test the existence of linear relations among $\mathbb{K}$ by testing whether $Rank(\mathbb{M})$ equals $Rank(A_0)$. If $Rank(A_2) \neq 0$, change the order of columns in A_2 to make sure that the order of variables is $(\mathbb{K}-\mathbb{K}_0, \mathbb{K}_0, c)$.

Algorithm 1. Pseudo-Code for Knowledge-Propagation Phase

```
 1: function PROPAGATE($\mathbb{X}$, $\mathbb{K}$, $\mathbb{M}$)
 2:     $Flag \leftarrow true$
 3:     while $Flag$ do
 4:         $Flag \leftarrow false$
 5:         $\mathbb{M} \leftarrow GJE_{|\mathbb{X}-\mathbb{K}|}(\mathbb{M})$
 6:         for all rows r in $\mathbb{M}$ do
 7:             if only one non-zero variable in the first $|\mathbb{X}-\mathbb{K}|$ columns then
 8:                 $Flag \leftarrow true$                                  ▷ situation 2
 9:                 if $S(x) \in \mathbb{X}$ then
10:                     Change columns for $x$ and $S(x)$ in $\mathbb{M}$
11:                     $\mathbb{K} \leftarrow \mathbb{K} \cup \{x, S(x)\}$
12:                     go to line 3
13:                 else
14:                     Change columns for $x$ in $\mathbb{M}$
15:                     $\mathbb{K} \leftarrow \mathbb{K} \cup \{x\}$
16:                     go to line 3
17:                 end if
18:             end if
19:             if case $(ii)$ of Lemma 2 happens in r then
20:                 $Flag \leftarrow true$                                  ▷ situation 3
21:                 Change columns for $x$ and $S(x)$ in $\mathbb{M}$
22:                 $\mathbb{K} \leftarrow \mathbb{K} \cup \{x, S(x)\}$
23:                 go to line 3
24:             end if
25:         end for
26:         for all pairs $(x, S(x))$ in $\mathbb{X}$ do
27:             if case $(i)$ of Lemma 2 happens in $\mathbb{M}$ then
28:                 $Flag \leftarrow true$                                  ▷ situation 3
29:                 Change columns for $x$ and $S(x)$ in $\mathbb{M}$
30:                 $\mathbb{K} \leftarrow \mathbb{K} \cup \{x, S(x)\}$
31:                 go to line 3
32:             end if
33:         end for
34:     end while
35:     return ($\mathbb{K}$, $\mathbb{M}$)
36: end function
```

Denote by $\mathbb{E}_{\mathbb{K}}$ the subspace of $\mathbb{K}$ spanned by the row vectors of A_2. Indeed, $\mathbb{E}_{\mathbb{K}}$ is the subspace $\mathbb{E} \cap \mathcal{V}(\mathbb{K})$ of $\mathcal{V}(\mathbb{K})$. By Gauss-Jordan Elimination, the linear relations among $\mathbb{K}_0$ can be known by block matrix. However, more information can be known by S-box operations.

If there exist an $x \in \mathbb{K} - \mathbb{K}_0$ and a linear combination $\mathscr{L}' \in \mathcal{V}(\mathbb{K}_0)$ such that $x + \mathscr{L}' \in \mathbb{E}_{\mathbb{K}}$, then one can get $S(x) = S(\mathscr{L}')$. Since $\mathscr{L}' \in \mathcal{V}(\mathbb{K}_0)$, $S(\mathscr{L}')$ can be also deduced by variables in $\mathbb{K}_0$. Add $S(\mathscr{L}')$ to $\mathbb{K}$ and $\mathbb{K}_0$ (also add a new column corresponding to $S(\mathscr{L}')$), and add a new row corresponding to $S(x) + S(\mathscr{L}')$ to A_2 at the same time (make sure the order of variables is still $(\mathbb{K} - \mathbb{K}_0, \mathbb{K}_0, c)$). The reason to do this is that if there is a linear combination of $\mathscr{L}'' \in \mathcal{V}(\mathbb{K}_0)$ such that $S(x) + \mathscr{L}'' \in \mathbb{E}_{\mathbb{K}}$, then $S(\mathscr{L}')$ and $\mathscr{L}''$ can form a linear relation we want. Besides, if there is a linear combination of $\mathscr{L}'' \in \mathcal{V}(\mathbb{K}_0)$ such that $S(x) + e_y \cdot y + \mathscr{L}'' \in \mathbb{E}_{\mathbb{K}}$ ($y \notin \mathbb{K}_0$), then $e_y \cdot y + S(\mathscr{L}') + \mathscr{L}'' \in \mathbb{E}_{\mathbb{K}}$. So y can be used to gain more information just as x. This can be also applied to $S(x)$. This step is called **new-variable-adding**.

Algorithm 2. Pseudo-Code for Relation-Derivation Phase

```
1: function DERIVATION($\mathbb{K}_0$, $\mathbb{K}$, $A_2$)
2:     $Flag \leftarrow true$
3:     Change the order of columns in $A_2$
4:     while $Flag$ do
5:         $Flag \leftarrow false$
6:         $\mathbb{A}_2 \leftarrow GJE_{|\mathbb{K}-\mathbb{K}_0|}(A_2)$
7:         for all rows r in $A_2$ do                    ▷ new-variable-adding
8:             if only one non-zero variable $x$ in the first $|\mathbb{K} - \mathbb{K}_0|$ columns then
9:                 $Flag \leftarrow true$
10:                if $x$ is input of S-box then
11:                    Let $S(\mathscr{L}')$ be a new variables
12:                    $\mathbb{K} \leftarrow \mathbb{K} \cup \{S(\mathscr{L}')\}$ and $\mathbb{K}_0 \leftarrow \mathbb{K}_0 \cup \{S(\mathscr{L})\}$
13:                    Add a new column for $S(\mathscr{L}')$ and a new row for $S(x) + S(\mathscr{L},)$
14:                    go to line 4
15:                else
16:                    Let $S^{-1}(\mathscr{L})$ be a new variables
17:                    $\mathbb{K} \leftarrow \mathbb{K} \cup \{S^{-1}(\mathscr{L}')\}$ and $\mathbb{K}_0 \leftarrow \mathbb{K}_0 \cup \{S^{-1}(\mathscr{L}')\}$
18:                    Add a new column for $S^{-1}(\mathscr{L}')$ and a new row for $S^{-1}(x) + S^{-1}(\mathscr{L}')$
19:                    go to line 4
20:                end if
21:            end if
22:        end for
23:    end while
24:    $RelationSet \leftarrow \emptyset$
25:    $A_2 \leftarrow GJE(A_2)$
26:    for all row r in $B_2$ do
27:        Derive relation from r
28:        Add this relation to $RelationSet$
29:    end for
30:    return $RelationSet$
31: end function
```

After the step above, a matrix as $\begin{array}{c} \begin{array}{cc} \overbrace{}^{|\mathbb{K}-\mathbb{K}_0|} & \overbrace{}^{|\mathbb{K}_0|} \end{array} \\ \begin{pmatrix} B_0 & B_1 \\ 0 & B_2 \end{pmatrix} \end{array}$ can be known, $Rank(B_2)$ linear independent relations among $\mathbb{K}_0$ can be recovered from B_2.

The pseudo-code of the relation-derivation phase is shown in Algorithm 2. The inputs are the outputs of Algorithm 1. The function returns a set of relations among the variables of $\mathbb{K}_0$.

3.3 A Tool for Bit-Oriented Ciphers

The key schedules of some block ciphers have operations on word-level (e.g., S-box) and bit-level (e.g., cyclic shift), such as PRESENT, LBlock and so on. This tool is slightly different from the tool for word-oriented ciphers since it has operations both on words and bits. It also consists of two phases: knowledge-propagation phase and relation-derivation phase.

In the knowledge-propagation phase, since S-box permutation treats b bits as a union, situations 1 and 3 of Sect. 3.2 are no longer suitable for bit-oriented ciphers. The following lemma is used to deal with this situation.

Lemma 4. *Let $S[w_0^I \cdots w_{b-1}^I] = [w_0^O \cdots w_{b-1}^O]$, where w_i^I and w_i^O are 1-bit variables, respectively. If the values in any b out of $2b$ input/output bits of one S-box are known, then the values in the other b bits are uniquely determined, and can be computed efficiently.*

This situation can be dealt with by Gauss-Jordan Elimination as follows.

Lemma 5. *Let $\mathbb{S}$ be a set of input and output bit-variables of one S-box. If the order of variables in $\mathbb{M}$ is $(\mathbb{X}-\mathbb{K}-\mathbb{S}, \mathbb{S}, \mathbb{K})$ and $GJE(\mathbb{M})$ can be represented as:*

$$\begin{array}{ccc} \overbrace{\qquad}^{|\mathbb{X}-\mathbb{K}-\mathbb{S}|} & \overbrace{\quad}^{|\mathbb{S}|} & \overbrace{\quad}^{|\mathbb{K}|} \\ D_0 & D_1 & D_2 \\ 0 & D_3 & D_4 \end{array}$$

then the bit-variables of $\mathbb{S}$ can be uniformly determined if and only if $Rank(D_3) \geq b - n_k$, where n_k is the number of bits in $\mathbb{S}$ which are already in $\mathbb{K}$.

It is easy to see that since $\mathbb{S}$ is the set of input and output bit-variables of one S-box, the entropy of these bits is b. $Rank(D_3) = b - n_k$ means $b - n_k$ linearly independent relations can be built among variables in $\mathbb{S}$ and $\mathbb{K}$, and these relations are enough to reduce the entropy to 0.

This property is used in Algorithm 1 to get more information from S-box instead of situation 1 and 3 of Sect. 3.2

In the relation-derivation phase, suppose the order of variables in $\mathbb{K}$ is $(\mathbb{K} - \mathbb{K}_0 - \mathbb{S}, \mathbb{S}, \mathbb{K}_0)$ and $GJE(\mathbb{A}_2)$ can be represented as:

$$\begin{array}{ccc} \overbrace{\qquad}^{|\mathbb{K}-\mathbb{K}_0-\mathbb{S}|} & \overbrace{\quad}^{|\mathbb{S}|} & \overbrace{\quad}^{|\mathbb{K}_0|} \\ E_0 & E_1 & E_2 \\ 0 & E_3 & E_4 \end{array}$$

If $Rank(E_3) \geq b - n_k$, since $b - n_k$ linearly independent relations are enough to reduce the entropy to 0, $2b - n_k - Rank(E_3)$ new functions with the form $x_i + \mathscr{L}'$ can be added to A_2, where $x_i \in \mathbb{S}$ is not a pivot element of E_3 and $\mathscr{L}'$ is a variable denoting how x_i can be known from $\mathbb{S}$ and $\mathbb{K}_0$. Add $\mathscr{L}'$ to $\mathbb{K}_0$ and $\mathbb{K}$ at the same time. This step is used to replace the new-variable-adding step of Algorithm 2.

If $Rank(E_3) > b - n_k$, since $b - n_k$ linearly independent relations are enough to reduce the entropy to 0, the other $Rank(E_3) - (b - n_k)$ relations can be used to filter the variables in $\mathbb{K}_0$. Use the variables in $\mathbb{K}_0$ to deduce these relations, only $2^{b-Rank(E_3)}$ of them can satisfy the S-box table. For example, if we can get $b + 1$ S-box input/output bit-variables from $\mathbb{K}_0$, we can get 1 bit relation among the variables in $\mathbb{K}_0$, and it is obliviously one key-bridge we want. This property is used to get key-bridges in Algorithm 2.

We apply our automatic search tool to the attacks on LBlock and TWINE in the following sections.

4 Applications to LBlock

4.1 Description of LBlock

LBlock is a lightweight 64-bit block cipher designed by Wu et al. in 2011 [25] and is based on a variant of Feistel Network. It supports key size of 80 bits and the total number of iterations is 32. The Feistel function of LBlock is made up of a key addition AK, an S-box layer S made up of 8 4-bit S-boxes and a nibble permutation P. LBlock's function design can be visualized in Fig. 1. The key schedule of LBlock is rather simple. The 80-bit master key WK_0 is stored in a key register and represented as $WK_0 = WK_0[0] \cdots WK_0[79]$. At round i, the leftmost 32-bit of current content of register is output as round key. The key schedule of round i can be shown as follows $(i = 1, \cdots 31)$:

$$\begin{aligned}
&WK_i \leftarrow WK_{i-1} \lll 29,\\
&WK_i[0-3] \leftarrow S_9(WK_i[0-3]),\\
&WK_i[4-7] \leftarrow S_8(WK_i[4-7]),\\
&WK_i[29-33] \leftarrow WK_i[29-33] \oplus [i]_2.
\end{aligned}$$

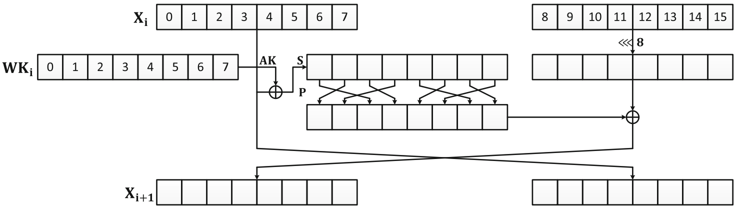

Fig. 1. Round function of LBlock block cipher

4.2 Impossible Differential Cryptanalysis on 23-Round LBlock

In INDOCRYPT 2012, Wu et al. presented an automatic search tool to search for the best impossible differential distinguishers [24]. In CRYPTO 2015, Sun et al. proved that this tool could find all impossible differentials of a cipher that are independent of the choices of the S-boxes [20]. In this paper, we automatize the search of the best impossible differential attacks by combining Wu's tool with our tool. Using Wu's tool, we can get all distinguishers with certain rounds. Using our key-bridging tool, we can get all the key-bridges to reduce the complexity in key-sieving phase.

Recently, Boura et al. [7] proposed a generic version of impossible differential attacks with the aim of simplifying and helping the construction and verification of this type of cryptanalysis. In particular, they provided a formula to compute the complexity of such an attack according to its parameters. To understand the formula, we first briefly review how an impossible differential attack is constructed. It starts by splitting the cipher into three parts: $E = E_3 \circ E_2 \circ E_1$ and finding an impossible differential ($\varDelta_X \nrightarrow \varDelta_Y$) through E_2. Then $\varDelta_X$ (resp. $\varDelta_Y$) is propagated through E_1^{-1} (resp. E_3) with probability 1 to obtain $\varDelta_{in}$ (resp. $\varDelta_{out}$). We denote by c_{in} and c_{out} the log_2 of the probability of the transitions $\varDelta_{in} \rightarrow \varDelta_X$ and $\varDelta_{out} \rightarrow \varDelta_Y$, respectively. Finally we denote by k_{in} and k_{out} the key materials involved in those transitions. All in all, the attack consists in discarding the keys k for which at least one pair follows the characteristic through E_1 and E_3 and in exhausting the remaining ones. The complexity of doing so is as follows:

- **data:** C_{N_α}
- **memory:** N_α
- **time:** $C_{N_\alpha} + (1 + 2^{|k_{in} \cup k_{out}| - c_{in} - c_{out}}) N_\alpha C_{E'} + 2^{|k| - \alpha}$

where N_α is such that $(1 - 2^{-c_{in} - c_{out}})^{N_\alpha} < (1 - 2^{-\alpha})$, C_{N_α} is the number of chosen plaintexts required to generate N_α pairs satisfying $(\varDelta_{in}, \varDelta_{out})$, $|k|$ is the key size and $C_{E'}$ is the ratio of the cost of partial encryption to the full encryption.

We use this framework to mount an impossible differential attack on 23-round LBlock. First we find an impossible differential distinguisher through 14 rounds of LBlock. The input (resp. output) inactive nibble of this distinguisher is at position 12 (resp. 5). It is extended by 4 rounds at the beginning and by 5 rounds at the end in order to attack 23 rounds of the cipher. It can be seen in Fig. 2 that the difference in the plaintexts has to be zero in 8 nibbles such that $c_{in} + c_{out} = 28 + 44 = 72$. The key material $k_{in} \cup k_{out}$ is composed of 36 round-key nibbles which can assume 2^{73} values thanks to our key-bridging tool. Specifically, we can find 71 linear independent key-bridges among these 36 round-key nibbles. We show parts of the key-bridges we found in the full version of this paper.

As a consequence, and according to the above formula, the memory complexity of our attack is $\alpha \cdot 2^{71.5}$, the time complexity is $\alpha \cdot 2^{73} \cdot C_{E'} + 2^{80-\alpha}$. As we estimate the ratio $C_{E'}$ to $36/184 \approx 2^{-2.4}$, the value of α minimizing the overall complexity is 6.8. So the memory complexity of our attack is $2^{74.3}$ bytes, the time complexity is $2^{74.5}$ 23-round LBlock encryptions and the data complexity is $2^{59.5}$ chosen plaintexts.

Besides this attack, we can get another 2 impossible differential attacks on LBlock with the same complexity, i.e., $((12, 0)5, 14, 4)$, $((12, 5)5, 14, 4)$, where $((l_a, l_b)R_b, R_d, R_e)$ means that the position of input (resp. output) inactive nibble of R_d-round distinguisher is l_a (resp. l_b), and the number of rounds before (resp. after) the distinguisher is R_b (resp. R_e).

4.3 Zero-Correlation Cryptanalysis on 23-Round LBlock

At ACISP 2014, Wang et al. gave a multidimensional zero-correlation linear attack on 23-round LBlock [22]. The main technique they used to improve the

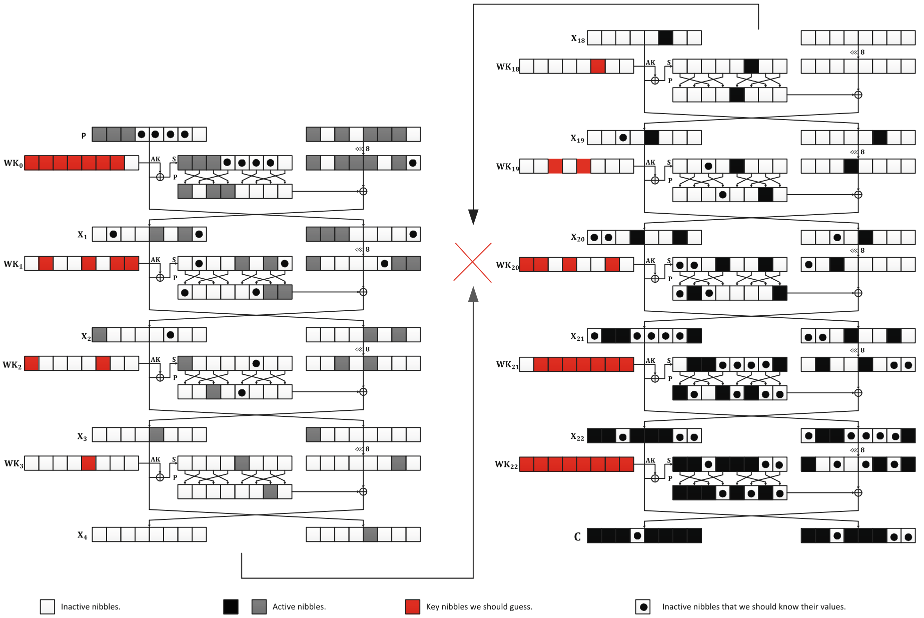

Fig. 2. Impossible differential attack on 23 rounds LBlock (Color figure online)

overall complexity is the partial compression technique, i.e., they reduce the complexity of online phase by guessing each sub-key nibble one after another. Since the time complexity of this attack is still greater than exhaustive search, they use 13 key-bridges to make this attack available.

According to their paper, for 14-round LBlock, if the input mask a of the first round locates at the left branch and the output mask b of the last round locates in the right branch, then the correlation of the linear approximation is zero, where $a, b \in F_2^4, a \neq 0$ and $b \neq 0$. Indeed, we find in total 21 key-bridges to reduce the overall complexity.

Combining this observation with our key-bridging tool, we find that $((1, 12)4, 14, 5)$ can get a better overall complexity. This is for the reason that we find 21 key-bridges thanks to our key-bridging tool. Since the major complexity of this attack comes from Step 4.1 and 4.2 of their paper, we explain the key-bridges of these 2 steps in detail.

The nibble X_4^1 corresponding to the input non-zero linear mask is affected by 32 bits of plaintext X_0 and 28 bits of round keys and the expression can be shown as:

$$X_4^1 = X_0^5 \oplus S(X_0^{12} \oplus S(X_0^0 \oplus WK_0^0) \oplus WK_1^2) \oplus S(X_0^{15} \oplus S(X_0^7 \oplus WK_0^7) \oplus \\ S(X_0^4 \oplus S(X_0^{10} \oplus S(X_0^1 \oplus WK_0^1) \oplus WK_1^0) \oplus WK_2^2) \oplus WK_3^3)$$

Similarly, the nibble X_{18}^{12} corresponding to the output non-zero linear mask is affected by 48 bits of plaintext X_0 and 48 bits of round keys:

$$X_{18}^{12} = X_{23}^{6} \oplus S(X_{23}^{12} \oplus WK_{22}^{4}) \oplus S(X_{23}^{15} \oplus S(X_{23}^{4} \oplus S(X_{23}^{13} \oplus WK_{22}^{5}) \oplus WK_{21}^{6}) \oplus WK_{20}^{1})$$
$$\oplus S(X_{23}^{12} \oplus S(X_{23}^{3} \oplus S(X_{23}^{10} \oplus WK_{22}^{2}) \oplus WK_{21}^{5}) \oplus S(X_{23}^{0} \oplus S(X_{23}^{9} \oplus WK_{22}^{1}) \oplus$$
$$S(X_{23}^{14} \oplus S(X_{23}^{2} \oplus S(X_{23}^{8} \oplus WK_{22}^{0}) \oplus WK_{21}^{4}) \oplus WK_{20}^{0}) \oplus WK_{19}^{4}) \oplus WK_{18}^{0})$$

Step 4.1. The guessed-keys of Step 4.1 are WK_0^1, WK_0^7, $WK_1^0[3]$, WK_0^0 and WK_{22}^1. Since $WK_0^7 \Rightarrow WK_1^0[0-2]$, WK_1^0 can be known. Meanwhile, since WK_{22}^1, $WK_{21}[33]$, $WK_{21}[34]$, $WK_{21}[35]$ and $WK_{21}[36]$ are inputs/outpus bits of one S-box, $WK_{21}[33]$, $WK_{21}[34]$, $WK_{21}[35]$ and $WK_{21}[36]$ can be known. Since $WK_{21}[34] \Rightarrow WK_{11}[4]$, $WK_{21}[35] \Rightarrow WK_{11}[5]$, $WK_{21}[36] \Rightarrow WK_{11}[6]$, $WK_0^1[0] \Rightarrow WK_{10}[34]$, $WK_0^1[1] \Rightarrow WK_{10}[35]$, $WK_0^1[2] \Rightarrow WK_{10}[36]$, $WK_0^0[3] \Rightarrow WK_{10}[33]$, and $WK_{11}[4]$, $WK_{11}[5]$, $WK_{11}[6]$, $WK_{10}[34]$, $WK_{10}[35]$, $WK_{10}[36]$, $WK_{10}[33]$ are input/output bits of one S-box, we can get 3-bit information to restrain the values of $WK_0^0[3]$, $WK_0^1[0-2]$ and WK_{22}^1. This is easily done by making a small lookup table.

As the following four equations:

$$X_1^5 = X_0^{15} \oplus S(X_0^7 \oplus WK_0^7),$$
$$X_2^2 = X_0^4 \oplus S(X_0^{10} \oplus S(X_0^1 \oplus WK_0^1) \oplus WK_1^0),$$
$$X_1^2 = X_0^{12} \oplus S(X_0^0 \oplus WK_0^0),$$
$$X_{22}^{10} = X_{23}^0 \oplus S(X_{23}^9 \oplus WK_{22}^1),$$

are true for LBlock, the 80-bit plaintext and ciphertext state value which affects the value of $X_4^1||X_{18}^{14}$ can be reduced to 60-bit after guessing the 14-bit equivalent key. The time complexity of this step is $N \cdot 2^{14} \cdot 5$ S-box accesses.

Step 4.2. The guessed-key of Step 4.2 is WK_{22}^0. Since $WK_0[2] \Rightarrow WK_{10}[32]$, $WK_0[1] \Rightarrow WK_{10}[31]$, $WK_0[0] \Rightarrow WK_{10}[30]$, $WK_{19}[11] \Rightarrow WK_{11}[3]$, $WK_{19}[10] \Rightarrow WK_{11}[2]$, $WK_{19}[9] \Rightarrow WK_{11}[1]$, $WK_{19}[8] \Rightarrow WK_{11}[0]$ and $WK_{10}[32]$, WK_{10} $[31]$, $WK_{10}[30]$, $WK_{11}[3]$, $WK_{11}[2]$, $WK_{11}[1]$, $WK_{11}[0]$ are input/output bits of one S-box, 3-bit information of $WK_{19}[11]$, $WK_{19}[10]$, $WK_{19}[9]$ and $WK_{19}[8]$ can be known. Meanwhile, $WK_{22}^1 \Rightarrow WK_{19}^2[3]$, $WK_{19}[11]$, $WK_{19}[10]$, $WK_{19}[9]$ and $WK_{19}[8]$ can be known.[2] Since $WK_{19}[9] \Rightarrow WK_{21}[32]$, $WK_{19}[8] \Rightarrow WK_{21}[31]$, $WK_{19}[7] \Rightarrow WK_{21}[30]$, and $WK_{22}[0]$, $WK_{22}[1]$, $WK_{22}[2]$, $WK_{22}[3]$, $WK_{21}[32]$, $WK_{21}[31]$, $WK_{21}[30]$ are input/output bits of one S-box, 3-bit information of WK_{22}^0 can be known. Since $X_{22}^{12} = X_{23}^2 \oplus S(X_{23}^8 \oplus WK_{22}^0)$, we can obtain a new state with 56-bit length. The time complexity of this step is $2^{60} \cdot 2^{14+1}$ S-box accesses.

The time complexity of the following sub-steps will become less and less thanks to our key-bridges. Let $N = 2^{62.1}$ as [22] shows, the time complexity of

[2] WK_0^0 and WK_{22}^1 are known from Step 4.1.

this attack is manipulated by Step 4.1, which is about $2^{62.1+14} \cdot 5 \cdot 1/8 \cdot 1/23 \approx 2^{71}$. The total time complexity is $2^{71} + 2^{71} = 2^{72}$ 23-round LBlock encryptions. The data complexity and memory complexity are the same as [22], i.e., the data complexity is $N = 2^{62.1}$ known plaintexts, the memory complexity is about 2^{60} bytes.

5 Applications to TWINE

5.1 Description of TWINE

TWINE is a lightweight 64-bit block cipher designed by Suzaki et al. in 2013 [21] and is based on a variant of Type-2 generalized Feistel structure. One version of TWINE uses an 80-bit key, another uses an 128-bit key and we denote these versions TWINE-80 and TWINE-128. The Feistel function of TWINE consists of an xor of a sub-key and a call to a unique S-box. TWINE's function design can be visualized in Fig. 3. The key schedule of TWINE-80 is quite simple. The 80-bit master key WK_0 is stored in a key register and represented as $WK_0 = WK_0^0 \cdots WK_0^{20}$.

The key schedule of round i can be shown as follows $(i = 1, \cdots 35)$:

$$\begin{aligned}
&WK_{i-1}^1 \leftarrow WK_{i-1}^1 \oplus S(WK_{i-1}^0), WK_{i-1}^4 \leftarrow WK_{i-1}^4 \oplus S(WK_{i-1}^{16}),\\
&WK_{i-1}^7 \leftarrow WK_{i-1}^7 \oplus 0||CONST_i^H, WK_{i-1}^{19} \leftarrow WK_{i-1}^{19} \oplus 0||CONST_i^L),\\
&WK_{i-1}^{0-3} \leftarrow WK_{i-1}^{0-3} \lll 4,\\
&WK_i \leftarrow WK_{i-1} \lll 16.
\end{aligned}$$

Then $WK_i^1||WK_i^3||WK_i^4||WK_i^6||WK_i^{13}||WK_i^{14}||WK_i^{15}||WK_i^{16}$ is used as the 8-nibble round key of round i. We use RK_i to denote the round key of round i. We refer to [21] for the 128-bit version of key schedule.

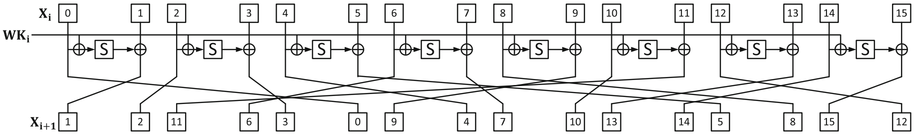

Fig. 3. Round function of TWINE block cipher

5.2 Zero-Correlation Cryptanalysis on TWINE

In [22], Wang et al. also gave multidimensional zero-correlation linear attacks on 23-round TWINE-80 and 25-round TWINE-128 using the partial compression technique.

However, using our automatic search tool, we find that the key-bridges they used in the attack on 23-round TWINE-80 do not exist. In their paper, they say that $RK_3^5 \Rightarrow RK_0^3$, $RK_2^6 \Rightarrow RK_0^1$ and $RK_{20}^1 \Rightarrow RK_{22}^6$. By the key schedule of TWINE-80,

$$\begin{aligned}
&RK_0^3 \Rightarrow WK_0^6 \Rightarrow WK_1^2 \Rightarrow WK_2^{17} \Rightarrow WK_3^{13} \Rightarrow RK_3^4,\\
&RK_0^1 \Rightarrow WK_0^3 \Rightarrow WK_1^{18} \Rightarrow WK_2^{14} \Rightarrow RK_2^5,\\
&RK_{20}^1 \Rightarrow WK_{20}^3 \Rightarrow WK_{21}^{18} \Rightarrow WK_{22}^{14} \Rightarrow RK_{22}^5.
\end{aligned}$$

So the key-bridges they used are not true.

Since RK_3^4 and RK_2^5 do not exist in the set of related round keys, this will make the time complexity of Step 4.1 in their paper greater than exhaustive search. So their attack on 23-round TWINE-80 is not available.

According to their paper, if the input mask a of the first round locates at the even nibble and the output mask b of the last round locates in the odd nibble for 14-round TWINE, then the correlation of the linear approximation is zero, where $a, b \in F_2^4, a \neq 0, b \neq 0$. Among these distinguishers, we find 4 of them with the minimal number of guessed-keys (with some key-bridges) by our automatic search tool. We use $((6,9)4,14,5)$ to get multidimensional zero-correlation linear attack on 23-round TWINE-80.

Three key-bridges we found are $RK_{22}^5 \Rightarrow RK_{20}^1$, $RK_{21}^0 \Rightarrow RK_{18}^4$ and $RK_{22}^6 \oplus S(RK_{22}^2) \oplus S(RK_{22}^5) \Rightarrow RK_2^6$.

Since the major complexity of this attack comes from Step 4.1 and 4.2 , we explain the key-bridges of these 2 steps. Assuming N known plaintexts are used.

Step 4.1. The distinguisher input nibble X_4^6 is affected by 32 bits of plaintext X_0 and 28 bits of round keys, and the distinguisher output nibble X_{18}^9 is affected by 48 bits of ciphertext X_{23} and 48 bits of round keys. Since $X_{22}^{11} = X_{23}^2 \oplus S(X_{23}^9 \oplus RK_{22}^5)$, X_{23}^9 and X_{23}^2 can be compressed to X_{22}^{11} by guessing RK_{22}^5. Since $X_{22}^{13} = X_{23}^{10} \oplus S(X_{23}^{15} \oplus RK_{22}^6)$, X_{23}^{15} and X_{23}^{10} can be compressed to X_{22}^{13} by guessing RK_{22}^6. Since $RK_{22}^5 \Rightarrow RK_{20}^1$, let $A = X_{23}^8 \oplus S(X_{23}^5 \oplus S(X_{23}^{12} \oplus S(X_{23}^7 \oplus RK_{22}^2) \oplus RK_{21}^0) \oplus RK_{20}^1)$, X_{23}^8, X_{23}^5, X_{23}^{12} and X_{23}^7 can be compressed to A by guessing RK_{22}^2 and RK_{21}^0. The time complexity of this step is $N \cdot 2^{16} \cdot 5$ S-box accesses.

Step 4.2. By guessing RK_{22}^2, one more nibble can be compressed. The time complexity of this step is 2^{64+16} S-box accesses.

Let $N = 2^{62.1}$, the time complexity of this attack is 2^{73} 23-round TWINE-80 encryptions, the data complexity is $2^{62.1}$, and the memory complexity is 2^{60} bytes.

For their attack on 25-round TWINE-128, we find another key-bridge besides their four key-bridges, i.e., $S(RK_{21}^1 \oplus S^{-1}(RK_0^3 \oplus RK_{24}^2)) \Rightarrow RK_{24}^3$. So after Step 4.3 of their attack, one more key nibble RK_0^3 can be known, and one more state nibble can be compressed in this step. The time complexity of the following steps is much smaller than the above 3 steps, so the time complexity of this attack is $(2^{62+60} \cdot 17 + 2 \cdot 2^{124})/(25 \times 8) + 2^{119} \approx 2^{120}$.

Besides, if ((12,3)5,14,6) is used to mount this attack, a better result can be got. Using our tool, four key-bridges can be found, i.e., $RK_0^3 \Rightarrow RK_3^1$, $RK_{21}^3 \Rightarrow RK_{24}^1$, $S(RK_{24}^0 \oplus S^{-1}(RK_2^5 \oplus RK_{23}^7 \oplus S(RK_{22}^2))) \Rightarrow RK_{22}^1$, $S(RK_{22}^3 \oplus S(RK_{19}^1)) \oplus S^{-1}(RK_0^7 \oplus RK_{24}^6 \oplus S(RK_{24}^7)) \Rightarrow RK_{22}^2$. The overall time complexity is determined by searching the remaining key candidates. So the time complexity becomes 2^{119}.

5.3 Impossible Differential and Meet-in-the-Middle Cryptanalysis on TWINE

At FSE 2015, Biryukov et al. gave impossible differential cryptanalysis and meet-in-the-middle cryptanalysis on 25-round TWINE-128 [2].

Combining Wu's tool for impossible differential distinguisher and our automatic tool, we find in total 12 attacks with the same time complexity as [2].

Combining Lin's propagate-then-prune tool for meet-in-the-middle distinguisher [19] and our automatic tool, we find some attacks with the same complexity as [2]. One of these attacks is $(6, 10) \rightarrow 5$, i.e., the distinguisher starts with two inactive nibbles at position (6,10) and ends with one nibble at position 5. This attack is useful when we want to get less false positive.

6 Conclusions

In this paper, we studied the key-bridging technique Dunkelman et al. proposed to deduce some sub-key bits from some other sub-key bits. We presented a versatile and powerful algorithm for searching key-bridging technique on word-oriented and bit-oriented block ciphers. This tool can not only give better attacked-rounds and complexity, but also a better understanding of the design of block ciphers. To demonstrate the usefulness of our approach, we used our tool to the impossible differential and multidimensional zero-correlation linear attacks on 23-round LBlock, 23-round TWINE-80 and 25-round TWINE-128. To the best of our knowledge, these results are the currently best results on LBlock and TWINE in the single-key setting.

Acknowledgements. We would like to thank the anonymous reviewers for providing valuable comments. The research presented in this paper is supported by the National Basic Research Program of China (No. 2013CB338002) and National Natural Science Foundation of China (No. 61272476, No. 61232009 and No. 61202420).

References

1. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). http://eprint.iacr.org/
2. Biryukov, A., Derbez, P., Perrin, L.: Differential analysis and meet-in-the-middle attack against round-reduced TWINE. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 3–27. Springer, Heidelberg (2015)

3. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
4. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
5. Bogdanov, A., Rechberger, C.: A 3-subset meet-in-the-middle attack: cryptanalysis of the lightweight block cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
6. Boura, C., Minier, M., Naya-Plasencia, M., Suder, V.: Improved Impossible Differential Attacks against Round-Reduced LBlock. Cryptology ePrint Archive, Report 2014/279 (2014). http://eprint.iacr.org/
7. Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to CLEFIA, Camellia, LBlock and Simon. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 179–199. Springer, Heidelberg (2014)
8. Daemen, J., Rijmen, V.: The Design of Rijndael: AES-the Advanced Encryption Standard. Springer, Berlin (2002)
9. Derbez, P., Fouque, P.-A., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 371–387. Springer, Heidelberg (2013)
10. Dunkelman, O., Keller, N., Shamir, A.: Improved Single-Key Attacks on 8-Round AES. Cryptology ePrint Archive, Report 2010/322 (2010). http://eprint.iacr.org/
11. Dunkelman, O., Keller, N., Shamir, A.: Improved single-key attacks on 8-round AES-192 and AES-256. In: Advances in Cryptology-ASIACRYPT 2010, pp. 158–176. Springer, Berlin (2010)
12. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
13. Hong, D., et al.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
14. Khovratovich, D., Biryukov, A., Nikolic, I.: Speeding up collision search for byte-oriented hash functions. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 164–181. Springer, Heidelberg (2009)
15. Knudsen, L., Leander, G., Poschmann, A., Robshaw, M.J.B.: PRINTcipher: a block cipher for IC-printing. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 16–32. Springer, Heidelberg (2010)
16. Ko, Y., Hong, S.H., Lee, W.I., Lee, S.-J., Kang, J.: Related key differential attacks on 27 rounds of XTEA and full-round GOST. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 299–316. Springer, Heidelberg (2004)
17. Li, L., Jia, K., Wang, X.: Improved single-key attacks on 9-round AES-192/256. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 127–146. Springer, Heidelberg (2015)
18. Li, R., Jin, C.: Meet-in-the-middle attacks on 10-round AES-256. In: Designs, Codes and Cryptography, pp. 1–13 (2015)
19. Lin, L., Wu, W., Wang, Y., Zhang, L.: General model of the single-key meet-in-the-middle distinguisher on the word-oriented block cipher. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 203–223. Springer, Heidelberg (2014)

20. Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., Alkhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 95–115. Springer, Heidelberg (2015)
21. Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: a lightweight block cipher for multiple platforms. In: Wu, H., Knudsen, L.R. (eds.) SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer, Heidelberg (2013)
22. Wang, Y., Wu, W.: Improved multidimensional zero-correlation linear cryptanalysis and applications to LBlock and TWINE. In: Susilo, W., Mu, Y. (eds.) ACISP 2014. LNCS, vol. 8544, pp. 1–16. Springer, Heidelberg (2014)
23. Wikipedia. Invariant Subspace –Wikipedia, The Free Encyclopedia (2015). https://en.wikipedia.org/wiki/Invariant_subspace
24. Wu, S., Wang, M.: Automatic search of truncated impossible differentials for word-oriented block ciphers. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 283–302. Springer, Heidelberg (2012)
25. Wu, W., Zhang, L.: LBlock: a lightweight block cipher. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 327–344. Springer, Heidelberg (2011)
26. Zheng, X., Jia, K.: Impossible differential attack on reduced-round TWINE. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 123–143. Springer, Heidelberg (2014)
27. Zhijie, C.: Higher Algebra and Analytic Geometry. Springer, Berlin (2001). (in Chinese)