Revisiting Key Schedule's Diffusion In Relation With Round Function's Diffusion

Jialin Huang, Xuejia Lai

Department of Computer Science and Engineering
Shanghai Jiaotong University, China

Abstract. We study the weakness of key schedules from an observation: many existing attacks use the fact that the key schedules poorly distribute key bits in the diffusion path of round function. This reminds us of the importance of the diffusion's relation between key schedule and round function. We present new cryptanalysis results by exploring such diffusion relation and propose a new criterion for necessary key schedule diffusion. We discuss potential attacks and summarize the causes for key schedules without satisfying this criterion. One major cause is that overlapping between the diffusion of key schedule and round function leads to information leakage of key bits. Finally, a measure to estimate our criterion for recursive key schedules is presented.
Today designing key schedule still lacks practical and necessary principles. For a practical key schedule with limited diffusion, our work adds more insight to its requirements and helps to maximize the security level.

1 Introduction

There is not a clear consensus on sufficient and necessary condition that a key schedule must satisfy. Previous designers paid more attention to the cipher algorithms. Key schedules have not achieved deserving attention and scrutiny as the algorithms. Many existing designing rules for key schedule are far away from practical guidance. Some of them are too strong to be used in practice; some of them are too weak to be secure enough; some of them only focus on a specific kind of attack and are one-legged; some of them are empirical and lack of sound reasons why they should be. However, more and more attacks using key schedule weaknesses indicate that the research on key schedule design principles is pressing. A poor key schedule can break a perfectly good cipher [34, 10, 35, 6].

Today we only know some basic principles. e.g. a key schedule should avoid (semi-)weak keys, equivalent keys, and complementation property [11]. Asymmetry such as round constants is also needed to prevent symmetry in the cipher leading to slide attacks, e.g. AES, Serpent. Totally independent and random subkeys are thought to be secure, but this requires a large number of key bits. Practical key schedule generates paseudorandom subkeys at most. One type of "strong" key schedule is constructed using one-way function, or encryption cipher to achieve weakly one-way. These key schedules are not adopted by nearly all widely used block ciphers due to their low efficiency lacking obvious necessity. Early key schedules are very simple, including following kinds: simple permutation of master key, e.g. DES, IDEA; direct or recursive linear transformation from master key. As the development of cryptanalysis technique, the designer began to add non-linear operations, such as S-boxes, into key schedules for avoiding related-key (differential) type attacks. Most of non-linear key schedules are falling into three types. First type is similar to the SPN structure of block cipher, such as AES. A typical case of second type is Serpent, which goes through a number of S-boxes after linear transformation. The third type usually uses cipher primitives to generate prekeys–this ensures a certain degree of confusion and diffusion, then transform prekeys into subkeys linearly. These key schedules are invertible to eliminate equivalent keys, which may appear in a one-way situation [17].

Our Contribution. In this paper, we observe that most of the attacks exploiting key schedule weakness are based on the fact that the key bits are insufficiently distributed in the diffusion of round function, and give examples to illustrate this fact. From this observation, we point out that the relation between the diffusion of cipher function and the diffusion of key schedule will affect the degree of encryption process's dependency on key bits. We discuss this relation in terms of the definitions of *calculation dependency path* and *actual key information.* A criterion for key schedule is proposed which is ignored by most designers now. Potential attacks when our criterion is not satisfied are discussed and causes are summarized. Moreover, by examining our criterion on other ciphers, new cryptanalysis results are presented, including: meet-in-the-middle attacks on 40-round SHACAL-2 and 25-round XTEA; improving a meet-in-the-middle attack on Serpent with lower time complexity and on AES with lower memory complexity; minor improvements for Safer++ and MISTY1. Finally, we present a way to measure our criterion for recursive key schedules.

Since most practical key schedules need high efficiency, their linear diffusion are limited and cannot achieve "completeness" as in most round functions. How to maximize the security level in this situation is meaningful. Our criterion concentrates on the linear layer of key schedules, and it is a necessary requirement whatever the non-linear components are.

2 Previous Work and Our Motivation

This section introduces previous design rules for key schedules. We highlight the work relating with linear transformation, and discuss what they lack of.

In [15], the author summarized the following purpose: "to present each key bit to a message input, and to an autoclave input, of each S-box as quickly as possible". By generating different permutations for DES key schedule, the author tested ciphertext dependence on key bits [31], and developed new principles: each key bit is used as input to each S-box in turn; no bit is used as autoclave inputs or excluded on successive rounds. From these empirical principles we can only derive that key bits should be uniformly used. More detailed effects caused by key schedules with different permutations and potential attacks caused by key schedules without satisfying above principles are not mentioned.

In [2], the author pointed out that all subkeys should be equally good, and defined a so-called "strong key schedule" with this property: it is "hard" to find any remaining key bits from any known key bits. The author then proposed a simple "strong key schedule" using encryption functions.

In [13], the author discovered that several practical attacks utilize the linear relation between subkey bits in different rounds, which avoids considerable amount of work. So he suggested this relation be practically intractable, and proposed two necessary properties: equal effect for all input key bits and weakly one-way. For these properties the author constructed a key schedule with n-folding algorithm and DES(3 single DES and 1 triple-DES), which slows down key setup obviously.

In [3] the author recommended maximizing avalanche in the subkeys. If possible every key bit should affect nearly every round in different ways. More specific principles are given in [4], including: hard to invert; no equivalent keys and collision-freedom–standard hash function properties; no dead spots; equally powerful effect of every key bit on the subkeys.

In [14], the author indicated that knowledge of one subkey provides no leakage of information about any other subkeys. Each subkey is generated by inputting all master key bits to a one-way function, ensuring the avalanche effect and the entropy.

In [11], the author adopted a invertible and recursive structure to design AES key schedule. He claimed that this structure must possess low implementation cost, sufficient diffusion and asymmetry.

In [12], the author stressed on one-wayness and minimal mutual information between all subkey bits and master key bits(using master key bits directly in subkeys leads to the worst case), which can be obtained by one-wayness. He measured one-wayness with Shannon's bit confusion and bit diffusion with statistical tests.

Above work can be summarized into two problems: no bit leakage and maximizing avalanche. First, key schedules should have no bit leakage in the level of "round", or of the whole key schedule. The former means they only consider leakage between different rounds of subkeys, or between some rounds of subkeys and master keys. When an recursive key schedule is invertible or when subkeys are direct transformation from master key, above leakage cannot be avoided. The latter means that no subkey bits can be derived easily whatever key knowledge is obtained. Many recent attacks show that the "round" consideration is not enough for security, while considering the whole key schedule is too strong with no attack supporting its necessity. So what kinds of leakage should a key schedule avoid? Moreover, the obscure understanding for the pattern of leakage leads to a recommendation of using one-way function, which has unpractical efficiency and existence of equivalent keys. Even the irreversibility of one-way function eliminates the leakage, it makes detecting equivalent keys difficult. For example, [12] applied AES round function to its key schedule for one-wayness, and this key schedule is proved to exist 2^{271} equivalent key pairs [17]. Second, maximizing avalanche effect reminds us of the importance of the diffusion in a key schedule. The AES designer claimed that his key schedule can achieve sufficient diffusion by taking a recursive structure. However, this key schedule has caused many attacks due to its slow diffusion, which is not the original intention of the designer. The definition of "sufficient" or "slow" for diffusion is unclear and just intuitive. So what requirements should the diffusion of key schedules satisfy? We consider above two problems together by relating the key schedule's diffusion with the cipher function's diffusion to propose a more detailed leakage pattern. Through our specific requirements, a key schedule can be designed more targeted.

3 Our Definitions

First, we improve a 6-round meet-in-the-middle attack [5] on Serpent [32] for explaining our definitions. The plaintext is B^0 and the ciphertext is B^6. B^i is the input of round i, X^i is the input of S-box layer in round i, and Y^i is the input of linear layer in round i. The bit to meet is $X^3_{3,7}$(at 3'th row, 7'th column of X^3). Fig. 1 shows the computation flow from the plaintext. The grey boxes represent the internal bits needed to compute in each round, and it is easy to know which key bits are involved in this computation.

The amount of the involved key is 237 bits by Fig. 1. However, according to the linear transformation in the key schedule, the grey 8 bits in K_2 can derive from the gray bits in K_0 and K_1. So actually the information of involved key is 229 bits, with a possible time complexity reduction. We check the decryption flow, and exchange the order of linear transform L in round i-1 and key XOR operation in round i. Instead of guessing K_4, and K_5, we guess $K_4^{'}$ and $K_5^{'}$, $K_i^{'} = L^{-1}(K_i)$. We need to guess 4 bits in $K_4^{'}$, 48 bits in $K_5^{'}$ and 124 bits in K_6. The total is 176 bits. So the total complexity depends on the computation from plaintext. The time complexity of original attack is 2^{247}, while our time complexity is 2^{233} with 2^5 data complexity. The improvement of our attack is mainly based on two facts. First is the existence of equivalent key steming from the characteristics of key-alternating structure, and this has been used in many other attacks such as on AES and ARIA. We stress on the second fact that the combined action of the key schedule's diffusion and the round function's diffusion makes the actual involved key bits less than their expected number. The effect of round function diffusion on the key schedule is not neglectable. One measure for diffusion is the completeness property, which deals with the dependency of each output bit on the input bits. So for the "incomplete" diffusion, we focus on the dependency of output bits on input bits and give the following definitions.

Definition 1. ***Calculation Dependency Path:*** *X is a m-bit intermediate value. According to the cipher round function, there is a calculation path for* X_0*:* $X_0 = f_1(X_1, K_1) = f_1(f_2(X_2, K_2), K_1) = f_1(f_2(f_3(X_3, K_3), K_2), K_1) = ... = f_1(...(f_s(X_s, K_s), K_{s-1}), ..., K_1)$*, Like Fig. 2.*

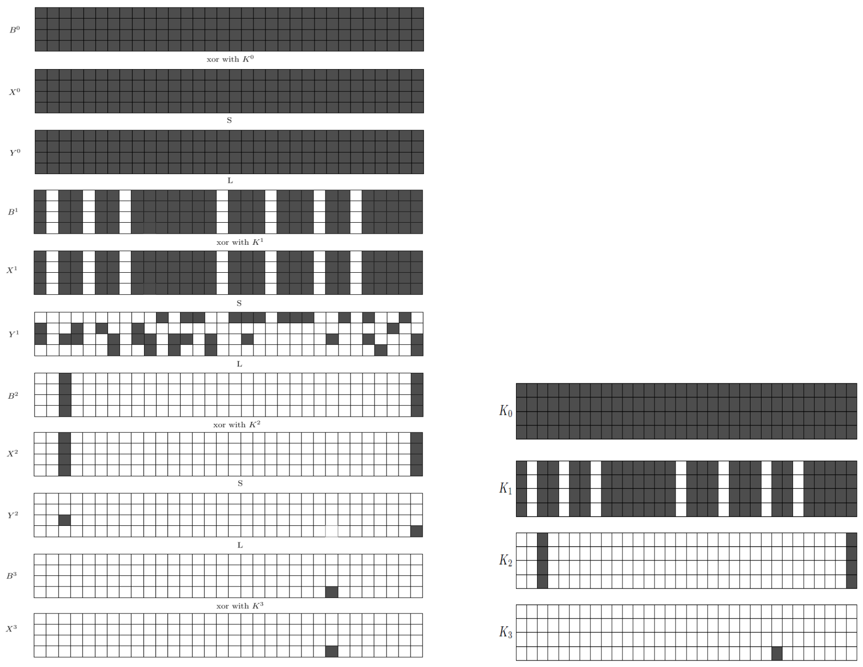

Fig. 1. The left is the 4-round calculation path of Serpent. The right is the corresponding 4-round involved subkeys.

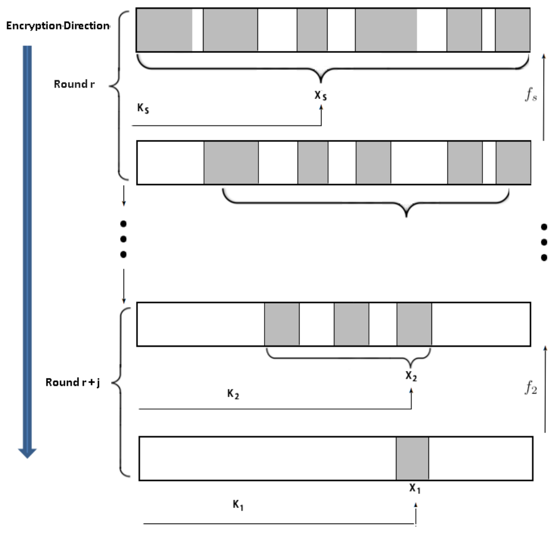

Fig. 2. A calculation dependency path. The grey parts are X_i. Note that the direction is considered as "backward", opposite to the encryption direction. Similarly, a "forward" calculation dependency path corresponding to decryption process is opposite to the decryption direction.

f_i is a sub-function depending on the cipher structure, where $i = 1, ..., s$. X_i is the set of input bits for f_i and X_{i-1} is the set of output bits, i.e. the calculation of X_{i-1} is dependent on X_i and K_i. $X \to X_1 \to X_2 \to ..... \to X_s$ is called a calculation dependency path.

Definition 2. ***Key Dependency Path:*** *The subkey bits involved in a calculation dependency path are denoted as a key dependency path. That is, $K_1, K_2, ..., K_s$.*

Definition 3. ***Actual Key Information(AKI):*** *According to a key schedule, the minimal amount of key bits which can easily derive all subkey bits in a key dependency path is called actual key information.*

The term "easily" depends on specific attack scenarios. We start the analysis from an output and trace its inputs successively, so the direction of path in Fig. 2 is considered as "backward", opposite to the encryption direction. Similarly, a "forward" calculation dependency path corresponding to decryption process is opposite to the decryption direction. The output can be one bit, or any size no larger than block size, depending on specific attack scenarios. A typical calculation dependency path is a one-round cipher involving one-round subkey. Obviously the diffusion layer mainly determines the related intermediate values in a path, since the non-linear module only has a limited range of diffusion(e.g. 4, 8, or 16-bit). Note that both calculation dependency path and its key dependency path depend on the diffusion of round function, and AKI is determined by the diffusion of key schedule. Ciphers with different structures have different forms of sub-functions. Take SPN structure as an example, the sub-function f_i can be easily known from the entries of linear transformation matrix corresponding to X_{i-1}. Look at Fig. 1, the grey boxes in the left show a calculation dependency path, and in the right show a key dependency path. X_0 is $X^3_{3,7}$, and X_s is B^0. The AKI is 229 bits of the linear map of master key.

In a combination of two or more paths, the definition of AKI still holds.

4 Existing Attacks Using Weakness of Key Schedules

One serious weakness of key schedules used by current attacks is the existence of calculation dependency paths missing some bits of master key('s map). The following examples illustrate how this insufficient AKI plays an important role in an attack[1].

4.1 IDEA

IDEA [26] is a 64-bit, 8.5-round block cipher with 128-bit keys. The input of round i is denoted by $X^i = (X^i_1, X^i_2, X^i_3, X^i_4)$. The output of KA layer in round i is $Y^i = (Y^i_1, Y^i_2, Y^i_3, Y^i_4)$. The subkey of round i is $Z^i = (Z^i_1, Z^i_2, Z^i_3, Z^i_4, Z^i_5, Z^i_6)$. p^i, q^i, s^i, t^i, u^i are intermediate values of MA layer in round i.

Recently, one of the most important cryptanalysis on IDEA is a meet-in-the-middle attack combining with the keyless Biryukov-Demirci relation [6]. The 6-round attack takes use of the following two calculation dependency paths. Backward: $s^4 \rightarrow p^4 \rightarrow (Y^4_1,Y^4_3) \rightarrow (X^4_1, X^4_3) \rightarrow (Y^3_1, t^3, Y^3_2, u^3) \rightarrow (Y^3_1, Y^3_2, p^3, q^3) \rightarrow (Y^3_1, Y^3_2, Y^3_3, Y^3_4) \rightarrow (X^3_1, X^3_2, X^3_3, X^3_4) \rightarrow (Y^2_1, Y^2_2, Y^2_3, Y^2_4)$. The key dependency path are: Z^4_1, Z^4_3, Z^4_5, Z^3_1, Z^3_2, Z^3_3, Z^3_4, Z^3_5, Z^3_6, Z^2_5, Z^2_6, with 112-bit AKI(the 50-33 bits of the master key). Forward: $s^5 \rightarrow p^5 \rightarrow (X^6_1, X^6_2) \rightarrow (Y^6_1, Y^6_2) \rightarrow (X^7_1, X^7_2, X^7_3, X^7_4) \rightarrow (X^8_1, X^8_2, X^8_3, X^8_4) \rightarrow (Y^8_1, Y^8_2, Y^8_3, Y^8_4)$. The involved subkeys are: Z^5_5, Z^6_1, Z^6_2, $Z^6_5, Z^6_6, Z^7_1, Z^7_2, Z^7_3, Z^7_4, Z^7_5, Z^7_6, Z^8_1, Z^8_2, Z^8_3, Z^8_4$, with 103-bit AKI(the 125-99 bits of the master key). Above AKI are both less than master key size(128-bit), which make a meet-in-the-middle attack possible. The leaked information between these two key dependency paths further reduces memory complexity[6].

Moreover, above 6-round attack is extended into 7.5-round with splice-and-cut technique, using another path [6]: $(X^2_1, X^2_2, Y^2_3, Y^2_4) \rightarrow (X^1_1, X^1_2, X^1_3, X^1_4)$. The combined AKI of this path(0-95 bits of the master key) and above forward path is only 103 bits, so the top and bottom rounds can be connected to target more rounds. Above backward path is modified as $s^4 \rightarrow (X^2_1, X^2_2, Y^2_3, Y^2_4)$, without changing its corresponding AKI.

[1] In the examples of this paper, we take the same notations as in the paper of original attacks.

Finally, [6] proposed the distributive technique for speeding up exhaustive search, which save the searching complexity for the missing key bits in a calculation dependency path. The path $s^3 \rightarrow p^3 \rightarrow (Y_1^3, Y_3^3) \rightarrow (X_1^3, X_3^3) \rightarrow (X_1^2, X_2^2, X_3^2, X_4^2) \rightarrow (X_1^1, Y_2^1, X_3^1, X_4^1)$ misses 16-24 bits of the master key, so the attacker can perform operations in this path without guessing these 9 bits, and then perform the rest operations for all of the key guesses. This technique still needs to go over all the possible keys, but performs less than a full encryption for each key.

4.2 KASUMI

KASUMI [27] is a 64-bit, 8-round block cipher with 128-bit keys. The plaintext $P = L_0||R_0$. Let L_i and B_i be the input and output of $i + 1$ round function respectively. X[i_1,...,i_2] denotes the (i_2 - i_1 + 1)-bit data from i_1'th bit to i_2'th bit of X. Each round key is made up of KL_i, KO_i and KI_i. Here, $KL_i = (KL_{i,1}, KL_{i,2})$, $KO_i = (KO_{i,1}, KO_{i,2}, KO_{i,3})$ and $KI_i = (KI_{i,1}, KI_{i,2}, KI_{i,3})$, which are linearly derived from the master key $K = (k_1, k_2,...,k_8)$.

In [7], the author proposed a meet-in-the-middle attack using initial structure and partial matching. The construction of the initial structure is based on a forward calculation dependency path: $R_0[16, ..., 31] \rightarrow (L_1[16, ..., 31], B_0[16, ..., 31]) \rightarrow (L_0, B_1)$. In addition, when L_0 is $0x0000ffff$, the output of FL has nothing to do with KL_1. So the involved subkeys are: KO_1, KI_1, KL_2, $KI_{2,2}$, $KO_{2,2}$, $KI_{2,3}$, $KO_{2,3}$ containing 7 16-bit words of master key except k_3. Similarly, choose L_2 as $0x0000ffff$, the output of FL has nothing to do with KL_3, so the backward path $L_3[16, ..., 31] \rightarrow (B_2[16, ..., 31], L_1[16, ..., 31]) \rightarrow (L_2, B_1)$ also exclude k_3. These two paths bring about a initial structure separating k_3 in the process of searching the total key.

Moreover, using the distributive technique in [6] also can get a factor of almost 2 speedups for exhaustive search on KASUMI. Due to the absence of k_4 and k_8 in the top and bottom rounds of paths, the part of encryption performed until the full key guess is thus slightly more than 4 rounds.

4.3 MISTY1

8-round MISTY1 [28] has a more complicate key schedule than KASUMI while other components and structure are similar. However, although this key schedule adds non-linear S-boxes, its diffusion still exists flaws. In [8] the author extended an integral attack to 6-round MISTY1 by applying a weakness of the key schedule. We show this weakness also comes from a calculation dependency path with insufficient AKI. This example illustrates more clearly about the concept of sub-function in Definition 1 due to the structure of MISTY1. FO_i denotes FO function in round i, and it includes $FI_{i,1}$, $FI_{i,2}$, $FI_{i,3}$, and two key XOR operations. $IL_i||IR_i$ is the 32-bit input of FO_i. Each FI is keyed by $AKO_{i,j}$ and $AKI_{i,j}$, where j = 1, 2, 3. $OL_i||OR_i$ is the output of FO_i, which is derived from $(ML_i \oplus AKO_{i,4})$ || $(MR_i \oplus AKO_{i,5})$. ML_i || MR_i is the output after $FI_{i,3}$. Before each odd round and after the last round, there are additional two

FL functions. $KL_{i,1}$ and $KL_{i,2}$ are subkeys in FL_i. The master key K = (K_1, K_2,...,K_8), and another 8 16-bit words are $K_i^{'} = FI_{K_{i+1}}(K_i)$.

This attack starts before the FL_3, FL_4 layer and ends at the end of the cipher. The plaintext is $P^{(1..32)} \mid\mid P^{(33..64)}$ The beginning value X_0 is the input of $FI_{4,1}$. $X^{(i_1..i_2)}$ denotes the (i_2 - i_1 + 1)-bit data from i_1'th bit to i_2'th bit of X. $D^{(1..32)}$ is the output of FL_3, $D^{(33..64)}$ is the output of FL_4.
$X_0 = AKO_{4,1} \bigoplus IL_4 = AKO_{4,1} \bigoplus D^{(33..48)} \bigoplus OL_3$
$= AKO_{4,1} \bigoplus D^{(33..48)} \bigoplus ML_3 \bigoplus AKO_{3,4}$
$D^{(33..48)} = FL_{KL_{4,1},KL_{4,2}}(P^{(33..64)})$
$ML_3 = FI_{AKI_{3,1}}(AKO_{3,1} \bigoplus IL_3) \bigoplus IR_3 \bigoplus FI_{AKI_{3,2}}(AKO_{3,2} \bigoplus IR_3)$
$IL_3 = FL_{KL_{3,1},KL_{3,2}}(P^{(1..32)})$
$IR_3 = FL_{KL_{3,1}}(P^{(1..32)})$
So the calculation dependency path is $X_0 \rightarrow (D^{(33..48)}, ML_3) \rightarrow (D^{(33..48)}, IL_3, IR_3) \rightarrow (P^{(1..32)}, P^{(33..64)})$, with key dependency path $AKO_{4,1} \bigoplus AKO_{3,4}$, $AKI_{3,1}$, $AKO_{3,1}$, $AKI_{3,2}$, $AKO_{3,2}$, $KL_{3,1}$, $KL_{3,2}$, $KL_{4,1}$, $KL_{4,2}$. Due to the bits leakage of the key schedule, when $KL_{3,2}$ and $KL_{4,1}$ have been guessed, $AKI_{3,1}$, $AKI_{3,2}$ is also known. This diminished AKI makes the attack successful. Another path used in [8] begins from the input of $FI_{4,2}$.

4.4 Camellia

In [18], the author gave a few observations on Camellia's key schedule [29], which can also be summarized as key information leakage in the calculation dependency paths. Take Camellia-128 as an example, the AKI of the combination of (K_8, $K_{9,2}$) and ($K_{16,2}$, K_{17}) is only 82 bits(K_i is the subkey of round function F in round i, $K_{i,j}$ is the j'th byte of K_i), which makes a 4 rounds extension on a 6-round impossible differentials feasible. The same situation also occurs in Camellia-192 and Camellia-256.

5 New Attacks Using Weakness of Key Schedules

By checking calculation dependency paths, we find new meet-in-the-middle attacks for 40-round SHACAL-2 and 25-round XTEA. In addition to Serpent mentioned in Section 3, we also improve previous attacks on AES and Safer++.

From now on, X[i_1] denotes the i_1'th bit of X, and X[i_1,...,i_2] denotes the (i_2 - i_1 + 1)-bit data from i_1'th bit to i_2'th bit of X.

5.1 SHACAL-2

SHACAL-2 [21] is composed of 64 rounds using a variable key length up to 512 bits. A 256-bit plaintext is divided into eight 32-bit words A, B, C, D, E, F, G, H. P_i is the input of round i, and P_{i+1} is the output of round i.
$P_i = A_i \mid\mid B_i \mid\mid C_i \mid\mid D_i \mid\mid E_i \mid\mid F_i \mid\mid G_i \mid\mid H_i$. The best single-key attacks for SHACAL-2 are differential-linear attack on 32-round [19] and differential-nonlinear attack on 33-round [20]. Our attack is for 0-39 rounds with 512-bit

keys. The plaintext is P_0, and the ciphertext is P_{40}. We find a forward and a backward calculation dependency paths with insufficient AKI both starting from $H_{20}[0]$. See Table. 1 and Table. 2.

Table 1. The Backward Calculation Dependency Path and Its Key Dependency Path

Round	**Involved Intermediate Bits**	**Involved Key Bits**
	$H_{20}[0]$	no
19	$G_{19}[0]$	no
18	$F_{18}[0]$	no
17	$E_{17}[0]$	no
16	$(D_{16}[0], E_{16}[0], E_{16}[6], D_{16}[11], E_{16}[25], F_{16}[0], G_{16}[0], H_{16}[0])$	$K_{16}[0]$
15	$(C_{15}[0], D_{15}[0,...,25], E_{15}, F_{15}[0,...,25], G_{15}[0,...,25], H_{15}[0,...,25])$	$K_{15}[0,...,25]$
14	$(B_{14}[0], C_{14}[0,...,25], D_{14}, E_{14}, F_{14}, G_{14}, H_{14})$	K_{14}
13	$(A_{13}[0], B_{13}[0,...,25], C_{13}, D_{13}, E_{13}, F_{13}, G_{13}, H_{13})$	K_{13}
12	$(A_{12}[0,...,25], B_{12}, C_{12}, D_{12}, E_{12}, F_{12}, G_{12}, H_{12})$	K_{12}
11	P_{11}	K_{11}
10	P_{10}	K_{10}
9	P_9	K_9
8	P_8	K_8
7	P_7	K_7
6	P_6	K_6
5	P_5	K_5
4	P_4	K_4
3	P_3	K_3
2	P_2	K_2
1	P_1	K_1
0	P_0	K_0

The AKI of the backward path is 507 bits. There are totally 529 bits in the forward path which should have cover all the 512 bits master key. However, the leakage reduce the AKI to only 508 bits. With the knowledge of $K_{24}[0,...,25]$, $K_{25}[0,...,25]$, $K_{26},...,K_{39}$, once we guess $K_{24}[26,...,31]$, according to the key schedule $K_{23}[0,...,25]$ and $K_{22}[0]$ can be derived.

With above two paths we can mount a basic meet-in-the-middle attack by checking if $H_{20}[0]$ computed from plaintexts and ciphertexts match. We choose 4 plaintexts and their ciphertexts. The time complexity for meet-in-the-middle phase is $2^{508} + 2^{509}$, and for searching phase is 2^{508}, totally 2^{510}.

SHACAL-2 was recommended to be one of the NESSIE selections, while SHACAL-1 was not selected due to some concerns of its key schedule. However, our analysis exposes that the key schedule of SHACAL-2 also has flaws.

Table 2. The Forward Calculation Dependency Path and Its Key Dependency Path

Round	**Involved Intermediate Bits**	**Involved Key Bits**
20	$H_{20}[0]$	$K_{20}[0]$
21	$(A_{21}[0], B_{21}[0], B_{21}[2], B_{21}[13], B_{21}[22], C_{21}[0], D_{21}[0], F_{21}[0], F_{21}[6], F_{21}[11], F_{21}[25], G_{21}[0], H_{21}[0])$	$K_{21}[0]$
22	$(A_{22}[0], B_{22}[0], B_{22}[2], B_{22}[13], B_{22}[22], C_{22}[0], C_{22}[2], C_{22}[13], C_{22}[22], D_{22}[0], E_{22}[0], F_{22}[0], F_{22}[6], F_{22}[11], F_{22}[25], G_{22}[0], G_{22}[6], G_{22}[11], G_{22}[25], H_{22}[0])$	$K_{22}[0]$
23	$(A_{23}[0], B_{23}[0], B_{23}[2], B_{23}[13], B_{23}[22], C_{23}[0], C_{23}[2], C_{23}[13], C_{23}[22], D_{23}[0], D_{23}[2], D_{23}[13], D_{23}[22], E_{23}[0], F_{23}[0], F_{23}[6], F_{23}[11], F_{23}[25], G_{23}[0], G_{23}[6], G_{23}[11], G_{23}[25], H_{23}[0], H_{23}[6], H_{23}[11], H_{23}[25])$	$K_{23}[0, ..., 25]$
24	$(A_{24}[0, ..., 25], B_{24}, C_{24}[0, ..., 25], D_{24}[0, ..., 25], E_{24}[0, ..., 22], F_{24}, G_{24}[0, ..., 25], H_{24}[0, ..., 25])$	$K_{24}[0, ..., 25]$
25	$(A_{25}[0, ..., 25], B_{25}, C_{25}, D_{25}[0, ..., 25], E_{25}[0, ..., 25], F_{25}, G_{25}, H_{25}[0, ..., 25])$	$K_{25}[0, ..., 25]$
26	$(A_{26}[0, ..., 25], B_{26}, C_{26}, D_{26}, E_{26}[0, ..., 25], F_{26}, G_{26}, H_{26})$	K_{26}
27	P_{27}	K_{27}
28	P_{28}	K_{28}
29	P_{29}	K_{29}
30	P_{30}	K_{30}
31	P_{31}	K_{31}
32	P_{32}	K_{32}
33	P_{33}	K_{33}
34	P_{34}	K_{34}
35	P_{35}	K_{35}
36	P_{36}	K_{36}
37	P_{37}	K_{37}
38	P_{38}	K_{38}
39	P_{39}	K_{39}
	P_{40}	

5.2 XTEA

XTEA [30] has 64-bit block size and 128-bit key size, which is extended from TEA [36]. The best 25-round result is a three-subset meet-in-the-middle with 9 known plaintexts and $2^{120.4}$ XTEA computations [37]. Other recent results include [9, 22]. Our 25-round attack has similar complexity with [37] but is much simpler. The attacks in [9, 37] and ours all make use of calculation dependency paths without enough AKI.

$L_i||R_i$ and $L_{i+1}||R_{i+1}$ is the input and output of round i. The master key is made of four 32-bit words K_0, K_1, K_2 and K_3. Our attack is from round 23 to round 47, with the plaintext $L_{23}||R_{23}$ and the ciphertext $L_{48}||R_{48}$. We compute $L_{36}[0]$ from the plaintexts and the ciphertexts, and check if they match.

We trace those bits needed to know during the calculation for $L_{36}[0]$ from plaintext. In order to compute $L_{36}[0]$, we need to compute $R_{35}[0]$; In order to compute $R_{35}[0]$, we need to compute $L_{34}[0], R_{34}[0], R_{34}[5]$, and so on. This path is illustrated in Appendix 1. By examining the key schedule, we find that $K_3[21, ..., 31]$ of master key are not used. Similarly, $K_0[21, ..., 31]$ are not used for computing $L_{36}[0]$ from ciphertext. The attack is as follows:

Algorithm 5.1: A MEET-IN-THE-MIDDLE ATTACK FOR 25-ROUND XTEA(P)

```
for each guess of $K_0[0, .., 20], K_1, K_2, K_3[0, .., 20]$
do
    for each guess of $K_0[21, ..., 31]$ compute $L^i_{36}[0]$ from $p^i$, $i = 0, .., 7$,
    store these $L^i_{36}[0]$ in a hash table
    for each guess of $K_3[21, ..., 31]$ compute $L^i_{36}[0]$ from $c^i$, $i = 0, .., 7$,
    check if there is a match in the table
    if there is a match in the table, choose as a candidate key.
Exhaustively search the candidate keys.
```

The time complexity of the meet-in-the-middle phase needs $2^{106+11+3} = 2^{120}$ encryption. There are $2^{128-8} = 2^{120}$ candidate keys left, so the searching phase needs 2^{120} encryption. The total time complexity is 2^{121}, and the data complexity is 8.

Both XTEA and SHACAL-2 have the following common attributes leading to calculation dependency paths that can be used for attacking: the size of one round subkey is much less than the master key size, and the diffusion of cipher round function is slow for distributing key bits. If the key schedule's diffusion overlaps the diffusion of cipher round function to some extent, the actual number of distributed key bits further decreases. Above two attacks are mainly based on observations of key schedules, without other complicated cryptanalysis techniques. The lesson learned is that in a calculation path of cipher function, the key schedule should involve key bits outside this path as quickly as possible. Moreover, if there are any kinds of available relations between existing paths, the block cipher will be more unsafe.

5.3 AES

AES has 128-bit block size and supports three key sizes: 128, 192, and 256 bits. It is byte-oriented, and the state is organized as a 4 $\times$ 4 table with one-byte entries. $K^{(r)}$ and $C^{(r)}$ denote the subkey and the ciphertext of r'th round. $\hat{K}^{(r)}$ = $MixColumns^{-1}(K^{(r)})$. $X_{i,j}$ denotes the byte of X at row i, column j.

Two calculation dependency paths both starting from $C_{1,1}^{(5)}$ are used. The forward is: $C_{1,1}^{(5)} \rightarrow (C_{1,1}^{(6)}, C_{2,1}^{(6)}, C_{3,1}^{(6)}, C_{4,1}^{(6)}) \rightarrow C^{(7)} \rightarrow C^{(8)}$, with key dependency path denoted as A: $\hat{K}_{1,1}^{(6)}$, $\hat{K}_{1,1}^{(7)}$, $\hat{K}_{2,4}^{(7)}$, $\hat{K}_{3,3}^{(7)}$, $\hat{C}_{4,2}^{(7)}$, $K^{(8)}$. The backward is: $C_{1,1}^{(5)}$ $\rightarrow$ ($C_{1,1}^{(4)}$, $C_{2,2}^{(4)}$, $C_{3,3}^{(4)}$, $C_{4,4}^{(4)}$) $\rightarrow$ ($M_{1,1}^{(4)}$, $M_{2,2}^{(4)}$, $M_{3,3}^{(4)}$, $M_{4,4}^{(4)}$)(M is the value after MixColumn), with key dependency path denoted as B: $K_{1,1}^{(4)}$, $K_{2,2}^{(4)}$, $K_{3,3}^{(4)}$, $K_{4,4}^{(4)}$, $K_{1,1}^{(5)}$. Another key path denoted as C is: $K_{1,1}^{(0)}$, $K_{2,2}^{(0)}$, $K_{3,3}^{(0)}$, $K_{4,4}^{(0)}$, $K_{1,1}^{(1)}$.
In [23], the author described a meet-in-the-middle attack on 8-round AES-256 with two parts. First is precomputing all possible mappings of $C_{1,1}^{(1)} \rightarrow C_{1,1}^{(5)}$. Then search key bytes in A and C to choose and encrypt a suitable plaintext set, and do a partial decryption on the ciphertext set.

Algorithm 5.2: A meet-in-the-middle attack for 8-round AES-256(P)

```
for each guess of $K_{3,3}^{(4)}, K_{4,4}^{(4)}, K_{1,1}^{(5)}$
do {
    for each guess of $c_0, .., c_{21}$
    do {
        precompute $f_{c_0,..,c_{21},K_{3,3}^{(4)},K_{4,4}^{(4)},K_{1,1}^{(5)}}(x), x = 0, ..., 255$
        Store the $2^{176}$ 256 bytes sequence in a hash table
    }
    for each guess of key bytes in path C
    do {
        Partially encrypt the set of $2^{32}$ plaintexts to get the intermediate values of $C^{(1)}$
        Choose 256 plaintexts that $C_{1,1}^{(1)}$ takes every possible value and the other bytes fixed
        Sort the 256 plaintexts chosen above indexed by their values of $C_{1,1}^{(1)}$
        for each guess of $K_{1,3}^{(4)}, K_{2,3}^{(4)}, K_{2,4}^{(4)}, K_{3,4}^{(4)}, K_{4,3}^{(4)}$,
        column 2 of $K^{(6)}$, column 1 of $K^{(8)}, \hat{K}_{1,1}^{(6)}, \hat{K}_{1,1}^{(7)}, \hat{K}_{2,4}^{(7)}, \hat{K}_{3,3}^{(7)}, \hat{K}_{4,2}^{(7)}$
        do {
            Deduce $K_{1,4}^{(4)}$ from $K_{1,1}^{(5)}, K_{1,3}^{(4)}, and K_{1,1}^{(1)}$
            Deduce column 2, 3, 4 of $K^{(8)}$
            Partially decrypt the ciphertexts corresponding to the 256 plaintexts
            to get the sequence of the intermediate values of $C_{1,1}^{(5)}$.
            If this sequence matches one of the $2^{176}$ sequences in the hash table,
            record the key as a candidate, otherwise, continue with another guess.
        }
    }
}
```

The precomputation needs to search subkeys in path B and other 20 parameters, 25 bytes totally: c_0,..,c_{21}, $K_{3,3}^{(4)}$, $K_{4,4}^{(4)}$, $K_{1,1}^{(5)}$.
We improve the original attack by observing that the AKI of combination of A, B and C is less than the total number of bytes involved, due to the leaked key bytes between A and B. By the key schedule of AES-256, knowledge of columns 1, 2, 3, 4 of the subkey $K^{(8)}$(belongs to key path A) allows to deduce columns 2,

3, 4 of the subkey $K^{(6)}$, columns 3, 4 of the subkey $K^{(4)}$, column 4 of $K^{(2)}$. So $K^{(4)}_{3,3}$, $K^{(4)}_{4,4}$ are leaked. Also, $K^{(3)}_{1,1}$ can derive from the knowledge of $K^{(1)}_{1,1}$(belongs to key path C) and $K^{(2)}_{1,4}$, then $K^{(5)}_{1,1}$ is also leaked from $K^{(3)}_{1,1}$ and $K^{(4)}_{1,4}$.

By above observation we can reduce the memory complexity by a factor of 2^{24} with a similar method mentioned in [6]. We rearrange the key bytes in the key search phase.
Choose a set of 2^{32} plaintexts with bytes on the diagonal taking all possible values and the other 12 bytes fixed. Obtain the corresponding ciphertexts. The attack is Algorithm 5.2.
The memory complexity of this attack is $2^{176} \times 256 = 2^{184}$ bytes, while the original attack is at least 2^{208} bytes. With a time-memory tradeoff, the original attack can reduce the memory complexity at the cost of increasing the time complexity. Our attack obtains a lower memory complexity without changing the time complexity–2^{208} encryptions. This is not the best result for 8-round AES-256, but can remind us of examining the key schedule more carefully.

5.4 Safer++

The author in [25] proposed integral cryptanalysis on 4-round Safer++$_{256}$ [33]. We find that a slight modification for this attack can bring another one round extension. K_{2r-1} and K_{2r} are subkeys in round r. $K^{(p)}_i$ denotes the p'th byte of K_i. A forward key dependency path is as follows: $K^{(1)}_7$, $K^{(1)}_8$, $K^{(5)}_9$, $K^{(6)}_9$, $K^{(7)}_9$, $K^{(8)}_9$, $K^{(9)}_9$, $K^{(12)}_9$, $K^{(5)}_{10}$, $K^{(6)}_{10}$, $K^{(7)}_{10}$, $K^{(8)}_{10}$, $K^{(9)}_{10}$, $K^{(12)}_{10}$. Another key path is: $K^{(3)}_1$, $K^{(4)}_1$, $K^{(6)}_1$, $K^{(7)}_1$, $K^{(9)}_1$, $K^{(10)}_1$, $K^{(13)}_1$, $K^{(16)}_1$, $K^{(3)}_2$, $K^{(4)}_2$, $K^{(6)}_2$, $K^{(7)}_2$, $K^{(9)}_2$, $K^{(10)}_2$, $K^{(13)}_2$, $K^{(16)}_2$. Although the combination of these two paths relates to 30 bytes subkeys, its AKI is only 22 bytes. This makes the 5-round attack successful: 2-round integral property and 3 rounds external extension. Unlike the original attack, we only check whether one byte is balanced and this is enough to be a non-trivial attack.

6 Our Criterion and Further Discussion

Considering above attacks, we propose a criterion for the design of key schedule: *A key schedule should have no r-round key dependency path with AKI less than the size of master key.* r is dependent on specific security standards for different block ciphers. For a high security level, r should be as small as possible.
We discuss the unsafe consequences when our criterion is not satisfied:

– If a backward/forward calculation dependency path of length r with insufficient AKI exists, then there is a r-round extension for an attack with a proper data complexity. e.g. the attacks in [24] and in Section 4.3.
– If a backward calculation dependency path of length r_1 and a forward calculation dependency path of length r_2 with insufficient AKI and certain kind of connection exist, then there is a meet-in-the-middle attack. e.g. the attacks in Section 5.1, 5.2. Memory complexity of this attack can be further

reduced when there is leakage in their key dependency paths. e.g. the attack in Section 5.3. The simplest connection is that the outputs of these two paths are the same one, which results in a basic meet-in-the-middle attack. If the connection is of r_3 length, there is a $(r_1+r_2+r_3)$-round attack.

- If the total AKI of the combination of paths at the top and bottom is insufficient, a basic splice-and-cut technique is feasible. e.g. the attack in [6].
- If there are calculation dependency paths with insufficient AKI in successive sub-functions, a construction for initial structure is feasible. e.g. the attack in Section 4.2.
- When a calculation dependency path has been used in an attack, the existence of missing key bits in segments of this path will further reduce the time complexity. That is, we generalize "The distributive technique" mentioned in [6] to other attack scenarios—the original technique is just as an optimization for exhaustive search. We divide a path into consecutive segments and for some segments the search computation for the missing key bits is saved. For example, we can reduce the time complexity of 6-round attack on MISTY1 in [8] at least by a factor of 2. We separate $K_4^{'}$ and K_6 from the key guessing process, since they are only used in $AKI_{3,2}$, $KL_{4,1}$ and $KL_{4,2}$. That is, when we compute $D^{(33..48)}$ from the plaintexts, we needn't to guess K_2, $K_8^{'}$, K_3, K_5. The attacks for XTEA in [9] and Section 5.2 can also be slightly improved by this technique. The longer the segment is and the less AKI the segment has, the more time complexity can be reduced. Although improvement brought by this technique is very minor, it still implies a potential defect for key schedules. Even it is uncertain whether there will be any kinds of disastrous attacks based on this defect, obviously many existing attacks are easier than what they were considered before.

There are three major causes for existing key schedules without satisfying our criterion. In order to explain more clearly, we give 3-round examples which are simple but typical.

- The subkey bits are poorly used and distributed by the round function, especially when the subkey size is much smaller than the block size and master key size. For the 3-round path in the left of Fig. 3, it is obvious that only part of sk_1 is involved, while sk_2 and sk_3 are not used and distributed.
- The subkeys do not increase the avalanche effect of the master key(e.g. simple permutation), so key dependency paths probably contain a small amount of key information. See the right of Fig. 3, in this situation the first cause is eliminated and each intermediate value is keyed. Suppose that the master key is $(K_1, K_2, ..., K_k)$, $sk_{1,1} = K_2$, $sk_{1,2} = K_1$, $sk_{1,3} = K_5$, $sk_{1,4} = K_3$, $sk_{1,5} = K_8$, $sk_{2,1} = K_3$, $sk_{2,2} = K_5$, $sk_{2,3} = K_8$, $sk_{3,1} = K_1$. Although there are 9 sub-blocks of subkeys in this path, their AKI is only 5 sub-blocks.
- The key information leakage decreases the AKI. In our context, the leakage means key bits that can be easily derived from the AKI in a key dependency path. Leakage usually occurs when the patterns of diffusion in the cipher round function and in the key schedule are overlapping. Also see the right of Fig. 3, in this situation the subkeys are not simple permutation from the

master key and have some avalanche effect. Suppose that $sk_{2,1} = sk_{1,1} \oplus sk_{1,2} \oplus sk_{1,3}$, $sk_{2,2} = sk_{1,2} \oplus sk_{1,4} \oplus sk_{1,5}$, $sk_{3,1} = sk_{1,5} \oplus sk_{2,1} \oplus sk_{2,3}$, the AKI is 6 sub-blocks.

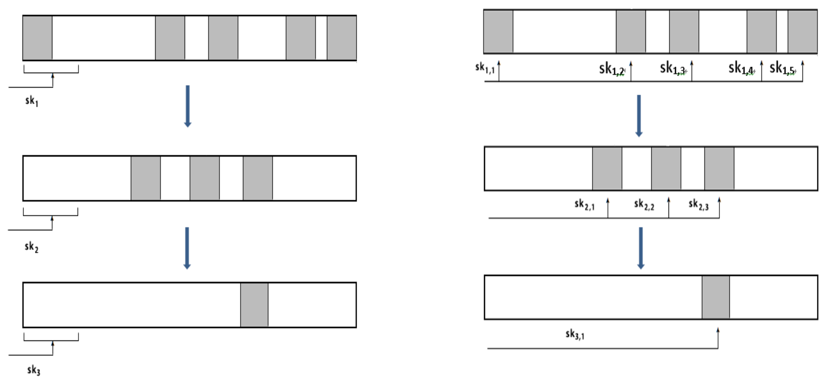

Fig. 3. Examples corresponding to different causes. sk_i means the subkey in round i. $sk_{i,j}$ means the j'th sub-block of sk_i which is involved in the calculation dependency path.

According to above causes, we suggest more detailed requirements for a key schedule. Every forward and backward key dependency path as well as their combinations should be carefully examined.

– The subkey size is no small than the block size and master key size, which makes sure every intermediate value in a path get keyed.
– More and more key bits should be involved with each transformation in the key schedule. A simple permutation, e.g. shift or rotation, is not preferred. Binary linear operations such as XOR and modulo addition are better.
– The diffusion path in the key schedule should be totally inconsistent with the diffusion in round function. That is, any subkey bits cannot be determined by the subkey bits that are in the same key dependency path.

7 Measure for Recursive Key Schedule

In this section we consider the requirements for linear transformation of recursive key schedules. A recursive key schedule means that the generation of current subkey is based on the value of previous subkeys. We believe that a recursive key schedule possesses more advantages in the design of block cipher: generating subkeys on-the-fly without precomputaion, which is especially requisite in hash mode application; increasing the avalanche effect of key bits round-by-round; security measure can be more normative and systematic. Recursive key schedules can help to control the speed and pattern of key bits diffusion by the iteration. We only need to investigate one round transformation and the results can extend to any rounds. If each subkey directly transforms from the master key without relations with its neighbour rounds, the measure is hard since we need to check key dependency paths from each round.

Consider the three causes we summarized in Section 6. The first cause is mainly due to the structure of block cipher and has nothing to do with key schedule's diffusion. Also, if the linear operations in recursive key schedules cannot involve more bits to increase the diffusion after each round (e.g, shift, rotation), we can analyze these simple key schedules directly and eliminate the second cause. Thus we only consider those recursive linear transformations with operations introducing more diffusion(e.g, XOR and modulo addition), and focus on avoiding the third cause: the diffusion of key schedule does not result in key information leakage in the diffusion path of round cipher. Note that this is only a necessary condition for a secure key schedule.

Usually linear transformations of both block cipher and key schedule can be represented as follows: $LT \to y_j = \sum a_{i,j}x_i + b_j$, where y_j and x_i are the output and input respectively, and $b_j \in \mathrm{GF}(2)$ is the constant coefficient. All $a_{i,j}$ over GF(2) constitute the representing matrix, and $a_{i,j} = 1$ means output j is dependent on input i. So we can know the input bits that each output bit depends on from each column of the matrix. M_1 represents the matrix of the round function, and M_2 represents the matrix of the key schedule. We discuss **the conditions M_2 should satisfy when M_1 is known**.

Denote k as master key size and n as block size. We take $k = 2n$ as an example, since this is common but typical when master key size larger than block size. For many block ciphers this situation is more vulnerable to attack than that $k = n$, so it deserves more attention. We consider consecutive 4 rounds of cipher, corresponding to 2 rounds of key scheduling. We take two rounds of cipher as one and the equivalent transformation matrix $M_1^{'}$ is $\begin{pmatrix} M_1^2 & M_1 \\ M_1^3 & M_1^2 \end{pmatrix}$, with the identity matrix rewritten as $I^{'} = \begin{pmatrix} I & 0 \\ M_1 & I \end{pmatrix}$. Now $M_1^{'}$, $I^{'}$ and M_2 have the same size $k \times k$. In our context, addition and multiplication are OR and AND respectively, and weight$(\cdot)$ means Hamming weight.
$M_2 = (\alpha_1, \alpha_2, ..., \alpha_k)$, α_i is the column vector of M_2.
$M_1^{'} = (\gamma_1, \gamma_2, ..., \gamma_k)$, γ_i is the column vector of $M_1^{'}$.
e_i is a k-bit vector, where i'th bit equal to "1" and other bits are "0", $i = 1, ..., n$.
For each $i, i \in [1..n]$:

Step.1 $\theta = I^{'} \cdot e_i = (\theta_1, \theta_2, ..., \theta_k)$.
Step.2 $\beta = M_1^{'} \cdot \theta = \sum_{j=1}^{k} \theta_j \gamma_j$, AKI = weight$(\beta)$

Step.3 For each $j, j \in [1..k]$, if weight($\theta_j \alpha_j + \beta$) $>$ weight(β), increase AKI by 1 bit, else make θ_j equal to 0. Check if AKI $= k$. If AKI $= k$, go Step.4, otherwise, M_2 is unsafe.

Step.4 For any subset s with elements chosen from θ_1, θ_2,...,θ_k, if weight($\sum_{\theta_j \in s} \theta_j \alpha_j + \beta$)-[2] weight($\beta$) $<$ weigth(s), AKI = AKI - weight(s) +[3] weight($\sum_{\theta_j \in s} \theta_j \alpha_j + \beta$) - weight($\beta$). If AKI $< k$, M_2 is unsafe.

By inverse of LT we can examine the other direction of key dependency paths. We show an example to explain above procedure in Fig. 4. Assume M_1 is 4×4 and M_2 is 8×8, i.e. $k = 8$, and $n = 4$. They are as follows:

$$\underset{M_1^2}{\begin{pmatrix} 1000 \\ 0111 \\ 1111 \\ 1111 \end{pmatrix}} \quad \underset{M_1^3}{\begin{pmatrix} 1000 \\ 1111 \\ 1111 \\ 1111 \end{pmatrix}} \quad \underset{I'}{\begin{pmatrix} 10000000 \\ 01000000 \\ 00100000 \\ 00010000 \\ 10001000 \\ 01010100 \\ 10010010 \\ 01110001 \end{pmatrix}} \quad \underset{M_1'}{\begin{pmatrix} 10001000 \\ 01110101 \\ 11111001 \\ 11110111 \\ 10001000 \\ 11110111 \\ 11111111 \\ 11111111 \end{pmatrix}} \quad \underset{M_2}{\begin{pmatrix} 10000011 \\ 01000001 \\ 01100000 \\ 01110000 \\ 00111000 \\ 00011110 \\ 10001100 \\ 10000111 \end{pmatrix}}$$

$$\underset{I}{\begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0001 \end{pmatrix}} \quad \underset{M_1}{\begin{pmatrix} 1000 \\ 0101 \\ 1001 \\ 0111 \end{pmatrix}}$$

Fig. 4. Matrices of round function diffusion and key schedule diffusion.

When $i = 1$, $\theta = (1,0,0,0,1,0,1,0)$, $\beta = (1,0,1,1,1,1,1,1)$, AKI $= 7$. For $j = 1$, $\theta_1 \alpha_1 = (1,0,0,0,0,0,1,1)$, $\theta_1 \alpha_1 + \beta = \beta$. For other j, we can check that $\theta_j \alpha_j + \beta = \beta$. So there is no j satisfying that weight($\theta_j \alpha_j + \beta$) $>$ weight(β), AKI still equals to 7 bits. This shows that M_2 is unsafe and we need not go Step.4.

8 Conclusion

This paper focuses on the diffusion of key schedules and proposes a criterion by the definition of AKI: every key dependency path corresponding to the calculation dependency path should have enough AKI–the actual required key

[2] Here is the decimal subtraction, similarly hereinafter.

[3] Here is the decimal addition.

information. We have shown several published attacks to explain our criterion, and presented new cryptanalysis results on SHACAL-2, XTEA, Serpent, AES and Safer++. We believe that this criterion can be used to improve attacks for other block ciphers by analyzing their key schedules. Many previous design rules of key schedules lack necessity or practicality, so it is hard to convince designers to follow. Our work gives a practical guidance for both the design and the cryptanalysis. Open problems include how to design efficient key schedules satisfying our criterion and how to develop new attacks based on weak key schedules with paths of insufficient AKI. Our work emphasizes that the interaction between the diffusion of cipher round function and the diffusion of key schedule should get more attention. An analysis for the linear transformation of recursive key schedules is given in our work, and more effective and generalized analysis tools can be further developed.

References

1. J. Kelsey, B. Schneiery, and D. Wagner, *Key Schedule Weaknesses in SAFER+*, Second AES Candidate Conference, 1999.
2. Lars R. Knudsen, *Practically secure Feistel ciphers*, FSE 1993, LNCS, Vol. 809, pages 211-221, Springer, 1994.
3. J. Kelsey, B. Schneier, and D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, Advances in Cryptology–CRYPTO'96, pages 237-251, Springer, 1996.
4. J. Kelsey, B. Schneier, and D. Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA*, INFORMATION AND COMMUNICATIONS SECURITY 1997, LNCS, Vol. 1334, pages 233-246.
5. T. Kohno, J. Kelsey, and B. Schneier, *Preliminary Cryptanalysis of Reduced-Round Serpent*, Third AES Candidate Conference, pages 195-211, 2000.
6. E. Biham, O. Dunkelman, N. Keller, and A. Shamir, *New Data-Efficient Attacks on Reduced-Round IDEA*, http://eprint.iacr.org/2011/417.
7. K. Jia, H. Yu, X. Wang, *A Meet-in-the-Middle Attack on the Full KASUMI*, http://eprint.iacr.org/2011/466.
8. X. Sun, X. Lai, *Improved Integral Attacks on MISTY1*, SAC 2009, LNCS, Vol. 5867, pages 266-280.
9. G. Sekar, N. Mouha, V. Velichkov, B. Preneel, *Meet-in-the-Middle Attacks on Reduced-Round XTEA*, Topics in Cryptology - CT-RSA 2011, LNCS, Vol. 6558, pages. 250-267. Springer 2011.
10. J. Kelsey, B. Schneier, *Key-Schedule Cryptanalysis of DEAL*, SAC '99 Proceedings of the 6th Annual International Workshop on Selected Areas in Cryptography pages 118-134.
11. J. Daemen, V. Rijmen, *The Design of Rijndael AES - The Advanced Encryption Standard*, 2002.
12. L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, *Strengthening the Key Schedule of the AES*, INFORMATION SECURITY AND PRIVACY, LNCS, 2002, Vol, 2384, pages 117-134.
13. U. Blumenthal, Steven M. Bellovin, *A Better Key Schedule For DES-LIKE Ciphers*, Proceedings of Pragocrypt 96.

14. G. Carter, E. Dawson, L. Nielsen, *Key Schedules of Iterative Block Ciphers*, INFORMATION SECURITY AND PRIVACY LNCS, 1998, Vol. 1438, pages 80-89.
15. L. Brown, *Key scheduling DES type cryptosystems*, Auscrypt 1990 Advances in Cryptography, pages 221-228, Springer 1990.
16. L. Brown, *A Proposed Design for an Extended DES*, Computer Security in the Age of Information, W. J. Caelli (editor), North-Holland, Amsterdam, 1989.
17. J. Choy, A. Zhang, K. Khoo, M. Henricksen, and A. Poschmann, *AES Variants Secure Against Related-Key Differential and Boomerang Attacks*, WISTP 2011, LNCS, Vol. 6633, pages 191-207, Springer, 2011.
18. J. Lu, Y. Wei, J. Kim, P.A. Fouque, *Cryptanalysis of Reduced Versions of the Camellia Block Cipher*, SAC 2011.
19. Y. Shin, J. Kim, G. Kim, S. Hong and S. Lee, *Differential-Linear Type Attacks on Reduced Rounds of SHACAL-2*, INFORMATION SECURITY AND PRIVACY, LNCS, 2004, Vol. 3108, pages 110-122.
20. Y. Wei, Y. Hu, J. Chen, *Differential-nonlinear attack on 33-round SHACAL-2* Journal of Xidian University 2010.
21. H. Handschuh, D. Naccache, *SHACAL : A Family of Block Ciphers*, Submission to the NESSIE project, 2002, http://www.cryptonessie.org.
22. A. Bogdanov, M. Wang, *Zero Correlation Linear Cryptanalysis with Reduced Data Complexity*, FSE 2012.
23. Hüseyin Demirci, and Ali Aydin Selçuk, *A Meet-in-the-Middle Attack on 8-Round AES*, proceedings of FSE 2008, LNCS, vol. 5086, pages 116-126, Springer, 2008.
24. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, *Improved cryptanalysis of Rijndael*, In B. Schneier, editor, FSE, LNCS, vol. 1978, pages 213-230, Springer, 2000.
25. G. Piret and J.J. Quisquater, *Integral Cryptanalysis on reduced-round Safer++*, IACR Cryptology ePrint Archive(2003)33-33.
26. X. Lai, J.L. Massey, and S. Murphy, *Markov Ciphers and Differential Cryptanalysis*, Advances in Cryptology, proceedings of EUROCRYPT 1991, LNCS 547, pages 17C38, Springer, 1992.
27. 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, 3G Security, Specification of the 3GPP Confidentiality and Integrity Algorithms; Document2: KASUMI Specification, V3.1.1 (2001).
28. M. Matsui, *New Block Encryption Algorithm MISTY*, FSE 1997, pages 54-68.
29. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, *Camellia: a 128-bit block cipher suitable for multiple platformsdesign and analysis*, In: Stinson, D.R., Tavares, S.E. (eds.) SAC 2000. LNCS, Vol. 2012, pages 39-56. Springer, 2001.
30. R.M. Needham, D.J. Wheeler, *Tea extensions, technical report*, Computer Laboratory, University of Cambridge, 1997, http://www.cix.co.uk/klockstone xtea.pdf.
31. C.H. Meyer, S.M. Matyas, *Cryptography: A New Dimension in Data Security*, John Wiley , Sons, New York, 1982.
32. R. Anderson, E. Biham, Lars R. Knudsen, *Serpent: A Proposal for the Advanced Encryption Standard*, NIST AES Proposal, 1998.
33. J.L. Massey, G.H. Khachatrian, and M.K. Kuregian, *Nomination of SAFER++ as Candidate Algorithm for NESSIE*, http://www.cryptonessie.org.
34. Lars R. Knudsen *A Detailed Analysis of SAFER K*, JOURNAL OF CRYPTOLOGY, Vol. 13, pages 417-436, 2000.
35. A Bogdanov, D Khovratovich, *Biclique cryptanalysis of the full AES*, ASIACRYPT 2011, LNCS, Vol. 7073, pages 344-371.

36. D.J. Wheeler, R.M. Needham, *TEA, a Tiny Encryption Algorithm*, FSE 1994, Vol. 1008, pages 363-366, Springer, 1994.
37. Y. Sasaki, L. Wang, Y. Sakai, K. Sakiyama, and K. Ohta, *Three-Subset Meet-in-the-Middle Attack on Reduced XTEA*, AFRICACRYPT 2012, LNCS, Vol. 7374, pages 138-154.

A Appendix 1

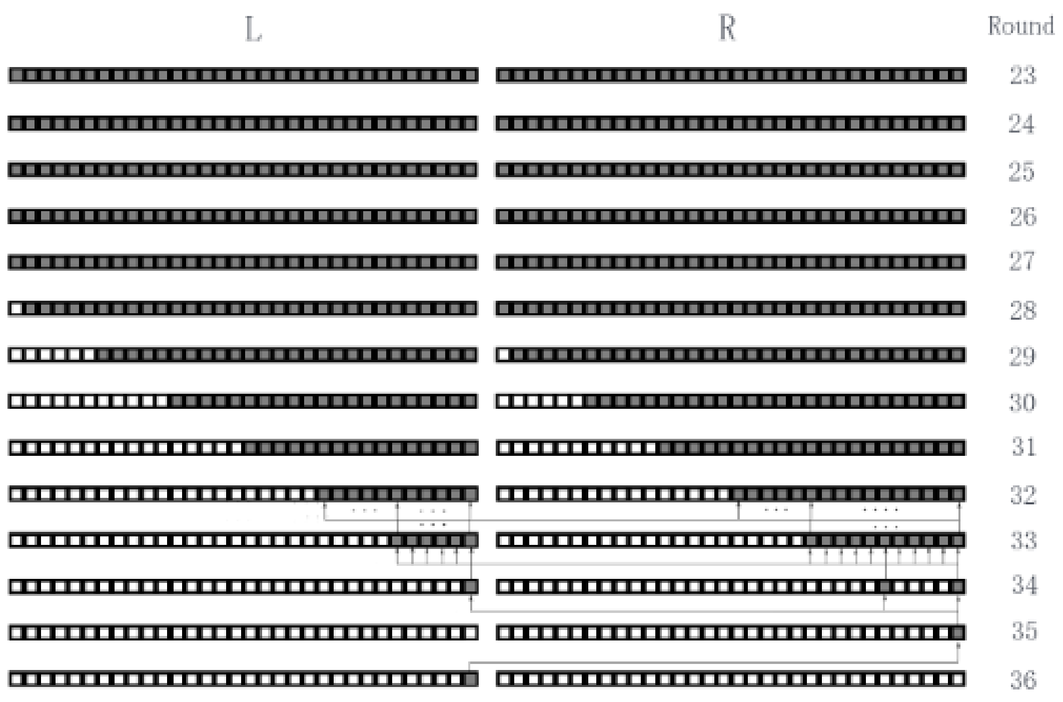

Fig. 5. The backward calculation dependency path for $L_{36}[0]$.