What is the Effective Key Length for a Block Cipher: an Attack on Every Block Cipher

Jialin Huang and Xuejia Lai

Shanghai Jiaotong University

Abstract. Recently, several important block ciphers are considered to be broken by the bruteforce-like cryptanalysis, with a time complexity faster than exhaustive key search by going over the entire key space but performing less than a full encryption for each possible key. Motivated by this observation, we describe a meet-in-the-middle attack that can always be successfully mounted against any practical block ciphers with success probability one. The data complexity of this attack is the smallest according to the unicity distance. The time complexity can be written as $2^k(1-\epsilon)$ where $\epsilon > 0$ for all block ciphers. Previously, the security bound that is commonly accepted is the length k of the given master key. From our result we point out that actually this k-bit security is always overestimated and can never be reached due to the inevitable key bits loss. No amount of clever design can prevent it, but increments of the number of rounds can reduce this key loss as much as possible. We give more insight in the problem of the upper bound of effective key bits in block ciphers, and show a more accurate bound. A suggestion about the relation between the key size and block size is given. That is, when the number of rounds is fixed, it is better to take a key size equal to the block size. Moreover, effective key bits of many well-known block ciphers are calculated and analyzed, which also confirm their lower security margin than thought before.

Keywords: block cipher, meet-in-the-middle, effective key length, exhaustive search

1 Introduction

As one of the fundamental primitives in symmetric cryptography, block ciphers play an important role in today's secure communication. They protest information data against unauthorized access and tampering in an insecure communication channel. Also, the design of many cryptographical schemes, such as secure encryption modes and authentication modes, is based on the security of block ciphers. Therefore, their security evaluation has been a hot research issue over the decades, giving rise to different analysis techniques. One line of research is the so-called provable security approach [12, 8], such as indistinguishability analysis. This approach usually studies design principles or cipher structures by assuming the pseudorandomness for some components. Another line of research

focuses on practical security, concerning that if any cryptanalytic attacks can be mounted successfully on a block cipher, such as differential attacks, linear attacks, meet-in-the-middle attacks, related-key attacks, as well as other existing cryptanalysis techniques. A block cipher is considered secure when it can resist against all known attacks. Traditionally the strength of a cryptanalytic attack is measured by comparing it to the exhaustive search over the entire key space. So the security of a block cipher mainly depends on the key length.

Recently the (full version of) AES has been called broken because of the biclique attack [4], which performs faster than the exhaustive search. In [9] the authors proposed a complex meet-in-the-middle attack on KASUMI using various subtle weaknesses of the cipher. In [3], the authors proposed several techniques to speed up exhaustive key search on the Full IDEA (by combining the BD-relation), KASUMI, and GOST. All of above attacks have the following in common: by going over the entire key space with performing less than a full encryption for each possible key, the full rounds of the ciphers are targeted with a time complexity slightly faster than exhaustive key search (for AES-256 is $2^{254.4}$, for KASUMI is $2^{125.8}$, for IDEA is $2^{126.8}$). These results are far from being any threat for the use of the ciphers in practice. However, they motivate us to consider the realistic complexity that an attack should be compared to. That is, in a real world context, what should the time complexity of a valid attack at most be.

1.1 Related Work

In [3], Biham et al recalled two well known techniques to marginally reduce the time complexity of exhaustive key search for almost any block ciphers. One is the distributive technique, which extracts the key bits that are not used in the first (or last) few operations of the encryption process. Another is the early abort technique proposed by [11], which is to discard a wrong key before computing the full ciphertext. Assume that a subset $K(1)$ of the key bits is not used in the first few operations, and a (possibly different) subset $K(2)$ is not used in the last few operations. Then Biham et al proposed a more advanced algorithm using the meet-in-the-middle technique, as follows.
For each value of the bits in $K \setminus K(1) \setminus K(2)$, perform the following:
1. For each value of the bits in $K(2) \setminus K(1)$, perform the first few operations of the encryption process for the given plaintext. Store the intermediate value and the corresponding value in $K(2) \setminus K(1)$ in a table.
2. For each value of the bits in $K(1) \setminus K(2)$, perform the last few operations in the decryption direction for the given ciphertext. Then guess the value of the remaining bits in $K(2)$, and complete the rest computation up to the intermediate value. Check the match with the values in the table.
Above algorithm (we call it Biham's algorithm in this paper) is enhanced further with the splice-and-cut technique by considering the common key bits that are not used in the operations between the plaintext and a pre-chosen intermediate value and in the last few operations, at the cost of increasing the data complexity (we call this splice-and-cut version of Biham's algorithm). Based on the cipher

structures and weaknesses of the key schedules, Biham et al showed the speedup for IDEA, GOST and KASUMI.

1.2 Our Contribution

Most block ciphers in common use are designed to have security equal to their key length (an exception is Triple-DES). Given that a key consists of k bits, the exhaustive search of the key space would take 2^k encryption, with success probability of one when the number of plaintext/ciphertext pairs satisfies the unicity distance.

In this paper, by giving a universal attack which has a time complexity of $2^k(1-\epsilon)$ where $\epsilon > 0$, we point out that the previously thought bound of the effective key size k can never be achieved for almost all practically used block ciphers. The data complexity of this attack is the smallest according to the unicity distance, and the success probability is about one. We present a formulated description, measuring the effective key length explicitly with some general parameters, such as block size, key size, and number of rounds. Moreover, our algorithm is applied to many well-known block ciphers and their effective key bits are calculated. As predicted, the effective key bits of these ciphers are all less than the master key size k.

Compared with previous work, our analysis is basing on more general structures and weaker assumptions, which have nothing to do with the specifics of key schedules. No matter how clever and secure the block cipher is, our algorithm is always available to the cryptanalyst. The data complexity of our algorithm reduces greatly. Only three instances are given for the splice-and-cut version of Biham's algorithm: IDEA, KASUMI, and GOST. This indicates that weak key schedules (all these three ciphers have simply linear key schedules) and large amount of data complexity are required for the attacks. No instances are given for the basic Biham's algorithm. We do a partial match in the middle, instead of using early abort technique in the ciphertext. More details about the algorithm such as the computational complexity are presented, not just a rough description. By the explicit quantization of the real bound of effective key lengths, the relation between key size and block size and the effect of increasing the number of rounds can be considered from a new point.

1.3 Organizations

This paper is organized as follows. In Section 2, we introduce the basic notations and construction for block ciphers. In Section 3, a generic attack is proposed and its computational complexities are studied. The upper bound of effective key bits is also investigated in this section. In Section 4, we give several widely used block ciphers as examples to show their effective key lengths. Section 5 discusses and concludes our results in this paper.

2 The Construction, Notations and Conventions

Based on Shannon's conception of confusion and diffusion, most modern block ciphers have been designed to use many iterations of substitution (nonlinear layer) and permutation (linear layer) to obtain enough security (each iteration is referred to as one round). We give the following notations first.

- P: plaintext where n is the block size
- C: ciphertext
- K: master key where k is the master key size
- R: the number of rounds
- S: the non-linear layer
- L: the linear layer
- K^r: the subkey used in round r
- X^r: the input block to round r where $X^0 = P$
- Y^r: the output block of the key mixing in round r
- Z^r: the output block of the nonlinear layer in round r

For almost all block ciphers used in practice, their R-round generic structure is depicted in Fig. 1.

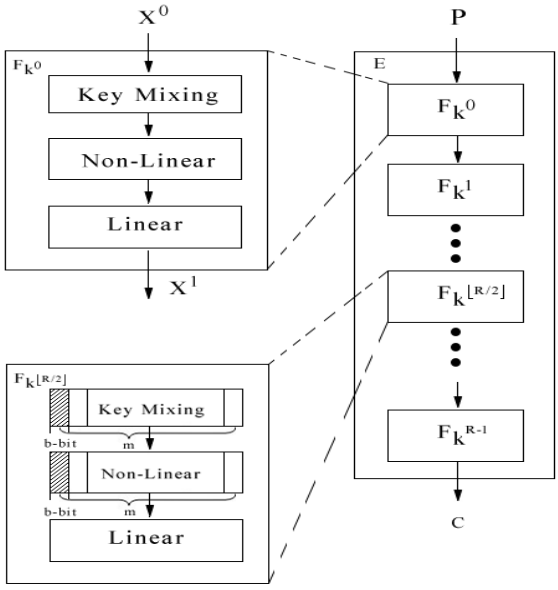

Fig. 1. Structure of an R-round block cipher E

There can be more than one non-linear or linear transformations in each round function. Usually the key mixing layer adds the subkey to the current

state block using linear operations, such as XOR and modulo addition. This means that the subkeys affect the round transformation with a separable pattern between different sub-blocks. That is, parts of subkeys directly act on parts of the internal state. e.g. K_i^r is mixed with X_i^r. Note that a round function in practice cannot be designed as random permutations, where the subkey can be regarded to act as a whole.

For a block cipher with key size k, the easiest and universal attack an adversary can mount is to simply try and guess each possible key. The probability of correctly guessing the key at the first attempt is 2^{-k}. Adding an additional bit to the length of key halves the probability that the key is correctly guessed. The time required to exhaust the whole key space is proportional to the time required to perform 2^k encryption operations.

3 A Generic Attack

We introduce a generic meet-in-the-middle attack that can be mounted on every practical block cipher. This attack is given in Algorithm 1.

Data: $\lceil \frac{k}{n} \rceil + 1$ pairs of plaintext and ciphertext
Result: the output key K
for *each value of the first* k_1 *key bits* **do**
  Compute S_1 from P with these k_1 bits ;
  for *each value of the remaining* $k - k_1$ *key bits* **do**
    Compute $Z_0^{\lfloor \frac{R}{2} \rfloor}$ from S_1;
    Store $Z_0^{\lfloor \frac{R}{2} \rfloor}$ in a table corresponding to the guessed key;
  end
end
for *each value of the last* k_1 *key bits* **do**
  Compute S_2 from C with these k_1 bits ;
  for *each value of the remaining* $k - k_1$ *key bits* **do**
    Compute $Z_0^{\lfloor \frac{R}{2} \rfloor}$ from S_2;
    if $Z_0^{\lfloor \frac{R}{2} \rfloor}$ *corresponding to the guessed key is in the table* **then**
      add the guessed key into the candidate list;
      move onto the next guess;
    else
      move onto the next guess;
    end
  end
end
Check keys in the candidate list with other $\lceil \frac{k}{n} \rceil$ plaintext/ciphertext;

Algorithm 1: The generic meet-in-the-middle attack

S_1 is the internal state that can be calculated from P only with k_1 bits of subkeys, where k_1 is the maximum smaller than k that can be obtained. Similarly, S_2 is the internal state that can be derived from C only with (other) k_1 bits of

subkeys. For any block cipher, the states of S_1 and S_2 certainly can be found. This algorithm has two phases, the meet-in-the-middle phase that generates the candidate list containing 2^{k-M} keys, and the check phase that examines the keys in the list.

For further discussion, we make two assumptions that are reasonable for practical block ciphers. First, non-linear transformations are assumed to consume much more time than linear transformations, so only non-linear operations are counted. Second, key schedules are assumed as negligible, since they are usually simpler than the encryption function.

Now we discuss the time complexity, data complexity, memory complexity and success probability for Algorithm 1.

Time complexity

Based on above assumptions, the time complexity is considered as follows. For any block ciphers, it is always smaller than 2^k.

$$\begin{aligned} T_{comp} = 2^{k_1}(\frac{N_{P\to S_1}}{N_{total}} + \frac{N_{C\to S_2}}{N_{total}} + 2^{k-k_1}\frac{N_{total} - N_{P\to S_1} - N_{C\to S_2} - N_{abort}}{N_{total}}) \\ +2^{k-M} + 2^{k-M-n} + 2^{k-M-2n}... \qquad (1) \\ \approx 2^k(\frac{N_{total} - N_{P\to S_1} - N_{C\to S_2} - N_{abort}}{N_{total}}) \\ = 2^k(1 - \frac{N_{P\to S_1} + N_{C\to S_2} + N_{abort}}{N_{total}}) \qquad (2) \end{aligned}$$

where N_{total} means the total non-linear components required in a full encryption. Denote $N_{P\to S_1}$ as the required non-linear components in the calculation from P to S_1. Denote $N_{C\to S_2}$ as the required non-linear components in the calculation from C to S_2. N_{abort} is the number of non-linear components needn't to be computed when partial matching techniques are used in the middle. The partial matching can filter M bits information of the key after the meet-in-the-middle phase. If write $\frac{N_{P\to S_1}+N_{C\to S_2}+N_{abort}}{N_{total}}$ as ϵ, then (2) is $= 2^k(1-\epsilon)$, where $\epsilon > 0$.

Equation (1) can be made more concrete for specific cipher structure. Usually there are two major structures for block ciphers, SPN and Feistel structure, as well as their generalized variants and combinations. Each n-bit internal state $W^r = (W_0^r, W_1^r,...,W_{m-1}^r)$ is a concatenation of m b-bit words W_i^r where b is the size of a non-linear sub-block (e.g. one S-box). For most SPN ciphers, every non-linear sub-block is keyed, and we match a b-bit word in the middle. So the time complexity is written as:

$$\begin{aligned} T_{comp} = 2^{k_1}(2\frac{(\frac{k}{b}-1)}{Rm} + 2^{k-k_1}\frac{Rm - (m - 1 + 2(\frac{k}{b}-1))}{Rm}) + 2^{k-b} + 2^{k-b-n} + 2^{k-b-2n}... \\ \approx 2^k(1 - \frac{m - 1 + 2(\frac{k}{b}-1)}{Rm}) \\ = 2^k(1 - \frac{1 - \frac{3b}{n} + \frac{2k}{n}}{R}) \qquad (3) \end{aligned}$$

where $m > 1$ in practical block ciphers due to limit of the size of one non-linear operation, as well as $k > b$. For an entire encryption, there are Rm non-linear operations. For the first $(\frac{k}{b} - 1)$ operations, we always do not need to guess all k bits of the key. We can compute S_1 by only searching the first $(k - b)$ bits, without guessing the remaining key bits. The time complexity of factor $(\frac{k}{b} - 1)$ is saved here. The multiplication of 2 means that the computation from both the plaintext and the ciphertext should be considered. In the middle round, using the partial matching technique, we can only compute one non-linear operation to get a b-bit filter and save another $(m - 1)$ operations.

For the Feistel structure, time complexity can be derived in the same way and the resulted formula is very similar. Due to the half diffusion property, at least one round of computation can be saved when matching in the middle. For other more detailed structures, such as MISTY (note that it has different sizes for non-linear components, 7 to 7 bits and 9 to 9 bits S-boxes) and Lai-Massey structures, we give examples directly in Section 4.

Data complexity

The required number of pairs of plaintext/ciphertext here is $U + 1$, where $U = \lceil \frac{k}{n} \rceil$ is the smallest data complexity according to the unicity distance. We use the first pair of data to filter parts of the wrong keys and generate the candidate list. Then we require at most another U pairs of plaintext/ciphertext for finding the right key.

If we store the internal states before and after the meet state in the middle, we can use the first data pair for filtering another $n - M$ bits. Now the first data also can provide all its n-bit information for checking, the same as other pairs of plaintext/ciphertext. So the data complexity can be reduce to U. Since the data complexity now is $U + 1$, which is small enough, this tradeoff is unnecessary.

Memory Complexity

Algorithm 1 has a memory complexity of $2^k \cdot M$ bits. If more memory can be sacrificed, the data complexity can be lowered as mentioned above. The time-memory tradeoff is not our concern here.

Success probability

In the meet-in-the-middle phase, a wrong key is eliminated with a probability of $1 - 2^{-M}$. When examined in the candidate list with the second data pair, a wrong key is discarded with a probability of $1 - 2^{-(M+n)}$. And when examined with the third data pair (if needed), a wrong key is eliminated with a probability of $1-2^{-(M+2n)}$, and so on. The success probability of Algorithm 1 is the product of these probabilities for all $2^k - 1$ wrong keys, which is approximately one.

Algorithm 1 is similar with Biham's algorithm, but has several differences. First, Biham's algorithm does not mention where the intermediate value is to meet. We explicitly claim that the meet position does not influence all the complexities of Algorithm 1. Without loss of generality, we fix this value as some sub-block in the middle round. Second, instead of aborting the evaluation after computing part of the ciphertext, we apply the early abort technique in the middle. That is, we partially match in the middle before computing the full intermediate state.

An Upper Bound of the Effective Key Length

No matter how clever and secure the block cipher is, Algorithm 1 is always available to the cryptanalyst. This indicates that a more accurate effective key length can be considered by taking the logarithm of the time complexity for this universal algorithm. As mentioned before, we can focus on (3).

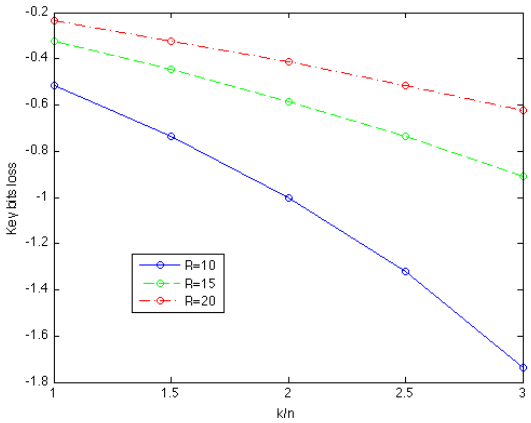

Fig. 2. The relation of key bits loss with key size, block size and number of rounds

The effective key size is $k+log(1-\frac{1-\frac{3b}{n}+\frac{2k}{n})}{R})$, which is always smaller than k for any block ciphers. Usually the size of non-linear sub-block b is much smaller than n and k, and only takes a few fixed values. For conventional block ciphers, the routine size of S-boxe is 4, 8, or 16 bits (for MISTY and KASUMI this is 7 or 9 bits). And for lightweight block ciphers, the routine size is 3 or 4 bits. $1-\frac{3b}{n}+\frac{2k}{n}$ is always larger than zero, so $log(1-\frac{1-\frac{3b}{n}+\frac{2k}{n})}{R}$ is smaller than zero. The previously accepted security bound of k key bits actually cannot be achieved.

Denote $log(1-\frac{1-\frac{3b}{n}+\frac{2k}{n})}{R})$ as the loss of effective key bits. We ignore the factor of $\frac{3b}{n}$ and draw the function of $log(1-\frac{1+\frac{2k}{n}}{R})$ with fixed different R. See Fig. 2. When R is the same, the larger $\frac{k}{n}$ is, the more key bits loss exists. When $\frac{k}{n}$ is the same, the more rounds a cipher iterates, the less key bits loss exists. So we can avoid the loss of key bits as much as possible by increasing the number of rounds or shortening the ratio of k and n. This indicates what a relation of key size and block size should be in a secure design, although is not very precise but still can be a rough guidance for most block ciphers. Also, from the formula we can conclude that when the number of rounds is sufficiently large, the key bits loss can very approximate to zero.

4 Effective Key Lengths for Block Ciphers

In this section, many practical block ciphers are analyzed for their actual effective key lengths. For a clear exhibition for (1), we consider conventional block ciphers and lightweight block ciphers respectively.

4.1 Conventional Block Ciphers

We take AES an an example, which is the most widely used block cipher now. It is selected as the new standard for replacing DES by NIST in 2000. As we assumed before, we count the computational complexity for S-boxes. This is also taken in [4]. For AES-128, the key size k and block size n are both 128 bits, the size of non-liner sub-blocks (S-box) b is 8, and the number of rounds R is 10. There are 16 sub-blocks in the state, that is $m = 16$. Refer to [6] for more details about AES. Note that the whitening key K^0 should also be considered here. The detailed application of Algorithm 1 is as follows. Z_0^5 is the intermediate value for meeting. Choose (Z_0^1, Z_1^1,...,Z_{14}^1) as S_1. Compute S_1 from P by guessing (K_0^0, K_1^0,...,K_{14}^0), 120 bits totally. Then for each guess of K_{15}^0, the last 8 bits of the master key K, complete the encryption operations from S_1 to Z_0^5. This requires a calculation of Z_{15}^1, Z^2, Z^3, Z^4, Z_0^5, 50 S-boxes totally. Store Z_0^5 in a hash table corresponding to the guessed key. Choose (X_1^{10}, X_2^{10},...,X_{15}^{10}) as S_2. Compute S_2 from C by guessing (K_1^{10}, K_2^{10},...,K_{15}^{10}), 120 bits again. Then for each guess of K_0^{10}, the last 8 bits of a mapping of the master key K, complete the decryption operations from S_2 to Z_0^5. This needs a computation of X_0^{10}, X^9, X^8, X^7, X^6, 65 S-boxes totally. There are $10 \times 16 = 160$ S-boxes for the full AES-128. So for each guess of 128-bit of K, only 115 S-boxes need to compute. This means $\frac{115}{160}$ of one full 10-round encryption for each guess. The time complexity of Algorithm 1 here is about $2^{128} \times \frac{115}{160} = 2^{127.5}$ (this value also can directly derive from (3)), so the effective key length can be regarded as 127.5 bits. We consider a little more about the structure of AES, that is, its branch number in the diffusion layer. Only four bytes knowledge of Z^4 is needed for computing Z_0^5, and four bytes knowledge of X^6 is needed. This can save additional 24 S-boxes, and the time complexity of Algorithm 1 is reduced to $2^{127.2}$. Similarly, the time complexity of Algorithm 1 for AES-256 is $2^{255.1}$. Compared with our upper bound of key bits, the best attack result so far on AES-256 with a time complexity of $2^{254.4}$ has much less gain than expected, since the effective key bits of AES-256 is actually only 255.1-bit.

We compute effective key lengths for other well-known block ciphers, listing in Table. 1.

We briefly explain for KASUMI [2]. Assume that the most time consuming sub-functions are three FI in each round for KASUMI. Only 7 16-bit words of the key require to be guessed before going to the third FI of round 1. Also, there is no need to guess all 128 bits of the key when the three FI operations are completed in round 8. Besides, the Feistel struture saves one more round in the middle, so there are 16 FI calculated for each guessed key. The time complexity is $2^{128} \times \frac{16}{24} = 2^{127.4}$.

Table 1. Time complexity of Algorithm 1 for conventional block ciphers, which also indicates effective key lengths

Block cipher	n	k	R	Time complexity of Algorithm 1	Previously best time complexity on full rounds
AES-128	128	128	10	$2^{127.2}$	$2^{126.1}$ [4]
AES-192	128	192	12	$2^{191.1}$	$2^{189.7}$ [4]
AES-256	128	256	14	$2^{255.1}$	$2^{254.4}$ [4]
SHACAL2	256	512	64	2^{511}	NO
MISTY1	64	128	8	$2^{127.6}$	NO
ARIA-128	128	128	12	$2^{127.4}$	NO
ARIA-128	128	192	14	$2^{191.4}$	NO
ARIA-128	128	256	16	$2^{255.3}$	NO
IDEA	64	128	8.5	$2^{127.4}$	$2^{126.1}$ [10]
KASUMI	64	128	8	$2^{127.4}$	$2^{125.8}$ [9]

Above ciphers are all recommended as standards or used by the industry for secure communications. According to our analysis, their security margin needs to be reconsidered. e.g. If an attack on SHACAL2 has a time complexity larger than 2^{511}, then this attack should be regarded as invalid. The best attack on full IDEA that was thought to optimize 1.9 bits now should be regarded as only 1.3 bits optimization.

4.2 Lightweight Block Ciphers

Most recently, secure communication on extremely constrained devices have been developing, such as RFID tags and sensor nodes. The constraints are mainly driven by cost and result in highly limited computing power, chip area and/or power supply, which mean that we must leave behind much of our conventional block ciphers. So the development of lightweight block ciphers is progressing greatly, resulting in more and more aggressive designs that often show two features. First, innovative techniques are used to improve existing ciphers. Second, the security margins that block ciphers are traditionally equipped with are reduced as much as possible in order to optimize the cipher performance. Due to these differences to conventional block ciphers, we discuss Algorithm 1 on lightweight block ciphers separately.

Take GOST as an example. GOST is known as the former Soviet encryption standard GOST 28147-89 which was standardized as the Russian encryption standard in 1989. It is well suitable for compact hardware implementations due to the simple structure, and the most compact implementation requires only 651 GE [13]. Therefore, GOST is considered as ultra lightweight. GOST has a 32-round Feistel structure with 64-bit block size n and 256-bit key size k. The F-function consists of eight S-boxes. Refer to [1] for more details. The application of Algorithm 1 is as follows. Due to the Feistel structure, we can check if R^{15} equals to L^{16}. Compute P to S_1 by guessing the 7 32-bit subkeys in the first seven rounds, and the least significant 28 bits of the subkey in round 8, 252

bits totally. Then for each guess of the most significant 4 bits of the subkey in round 8, complete the encryption from S_1 to R^{15}. This requires a calculation of 6 rounds and the last S-box in round 8, 49 S-boxes totally. Store the first 4 bits of R^{15} in a hash table corresponding to the guessed key. Similarly, compute C to S_2 by guessing the 7 32-bit subkeys in the last seven rounds, as well as the least significant 28 bits of the subkey in round 25, 252 bits totally. Then for each guess of the most significant 4 bits of the subkey in round 25, complete the decryption operations from S_2 to L^{16}. This needs a computation from round 16 to round 24, and the last S-box in round 25, 73 S-boxes totally. So for each guess of the 256-bit master key, 122 S-boxes require to be computed, which is $\frac{122}{256}$ of a full 32 rounds encryption (There are $8 \times 32 = 256$ S-boxes for the full GOST). The time complexity of Algorithm 1 is about $2^{256} \times \frac{122}{256} = 2^{254.9}$. So the effective key length is 254.9-bit. Moreover, we can only match part of R^{15} with part of L^{16}, e.g. their least significant 4 bits. In order to compute these 4 bits of R^{15}, only two S-boxes require to be calculated in round 14. Similarly, only two S-boxes are needed in round 16 for the matched 4 bits of L^{16}. 12 S-boxes are saved now, so the time complexity is slightly reduced to $2^{254.8}$. Note that, previous attacks on full GOST take use of its self-similarity property and relatively simple key schedule. We only consider the basic structure, which means that even the key schedule is much more complicated, Algorithm 1 still cannot be avoided.

Other results of lightweight block ciphers are summarized in Table 2. Some lightweight block ciphers have no non-linear components, e.g. XTEA. In this situation, different linear operations in the round function are considered to cost the same time, or we can simply take the round function as a unit when computing the time complexity.

Table 2. Time complexity of Algorithm 1 for lightweight block ciphers, which also indicates effective key lengths

Block cipher	n	k	R	Time complexity of Algorithm 1	Previously best time complexity on full rounds
GOST	64	256	32	$2^{254.8}$	2^{224} [7]
PRESENT-80	64	80	31	$2^{79.7}$	NO
PRESENT-128	64	128	31	$2^{127.6}$	NO
KATAN/KTANTAN	32/48/64	80	254	$2^{79.4}$	$2^{75.2}$ [5]
HIGHT	64	128	32	$2^{127.1}$	NO
XTEA	64	128	64	$2^{127.7}$	NO
Piccolo-80	64	80	25	$2^{79.7}$	NO
Piccolo-128	64	128	31	$2^{127.6}$	NO

5 Discussion and Conclusion

Recently, there are significant improvements on meet-in-the-middle attacks, as well as other bruteforce-like cryptanalysis. This makes us consider a universal

attack on all block ciphers, except the traditional exhaustive search method. In a practical cryptographic primitive, there are always some independent sub-modules. Computing these sub-modules with a independent pattern, instead of a combinational pattern, will save the overall time complexity. We describe a generic meet-in-the-middle attack that can always be mounted against any practical block ciphers. No amount of clever design can prevent it, no matter how many rounds or how complicated key schedule the cipher has. Note that having a sufficiently many rounds is still an important and expedient way to protect against it, since a larger number of rounds brings a higher complexity for Algorithm 1. We indicate a more accurate upper bound of effective key lengths for practical block ciphers, and claim that no these ciphers can reach their expected security margin, the given length of their master keys. Previously, exhaustive key search is generally considered as the benchmark with which other attacks are measured. A theoretical break (or academic break) against a block cipher is an attack with time complexity less than that of exhaustive key search, i.e. 2^k. Our analysis shows that tiny sacrifice of key bits is inevitable. So if an attack has the computational complexity larger than Algorithm 1 (even still faster than exhaustive search), it cannot be regarded as a valid attack. Algorithm 1 is also used to many well-known block ciphers and their effective key lengths are calculated. As predicted, the effective key bits of these ciphers are all less than the master key size k. However, our attack will not create a real threat for existing block ciphers, due to its limit caused by having to perform at least one operation for each possible key.

Another interesting discussion is about the relation between the block size with the master key size. Shannon's work on information theory shows that to achieve the perfect secrecy, it is necessary for the key size to be at least as large as the block size. That is, $k \geqslant n$. According to our analysis in Section 3, when the number of rounds is fixed, the larger $\frac{k}{n}$ is, the more loss of effective key bits there is. So $k = n$ is the best solution in the block cipher design in this context.

In the exhaustive key search, having to go through the entire key space before finding the correct key would be very unlucky, while being correct on the first guess would be very lucky. So the expected time to recover a k-bit key is 2^{k-1} encryptions. Note that most of the effective key lengths we calculate for existing block ciphers are larger than this average case, although some are still smaller. Given that the time complexity of Algorithm 1 in this paper is for the worst case, considering the average case and then comparing the result with 2^{k-1} can be the further work.

References

1. National soviet bureau of standards. information processing system - cryptographic protection - cryptographic algorithm gost 28147-89. 1989.
2. 3rd generation partnership project, technical specification group services and system aspects, 3g security, specification of the 3gpp confidentiality and integrity algorithms; document 2: Kasumi specification, v3.1.1. 2001.

3. E. Biham, O. Dunkelman, N. Keller, and A. Shamir. New data-efficient attacks on reduced-round idea. Cryptology ePrint Archive, Report 2011/417, 2011.
4. A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique cryptanalysis of the full aes. In *Proceedings of the 17th international conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'11, pages 344–371, Berlin, Heidelberg, 2011. Springer-Verlag.
5. A. Bogdanov and C. Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 229–240. Springer, 2010.
6. J. Daemen and V. Rijmen. Aes proposal: Rijndael. In *First Advanced Encryption Standard (AES) Conference*, 1998.
7. I. Dinur, O. Dunkelman, and A. Shamir. Improved attacks on full gost. In A. Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 9–28. Springer Berlin Heidelberg, 2012.
8. S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. In H. Imai, R. Rivest, and T. Matsumoto, editors, *Advances in Cryptology ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer Berlin Heidelberg, 1993.
9. K. Jia, H. Yu, and X. Wang. A meet-in-the-middle attack on the full kasumi. Cryptology ePrint Archive, Report 2011/466, 2011.
10. D. Khovratovich, G. Leurent, and C. Rechberger. Narrow-bicliques: Cryptanalysis of full idea. In D. Pointcheval and T. Johansson, editors, *Advances in Cryptology C EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 392–410. Springer Berlin Heidelberg, 2012.
11. J. Lu, Y. Wei, J. Kim, and P. Fouque. Cryptanalysis of reduced versions of the camellia block cipher. In *Selected Areas in Cryptography*, 2011.
12. M. Luby and C. Rackoff. How to construct pseudo-random permutations from pseudo-random functions. In H. Williams, editor, *Advances in Cryptology CRYPTO 85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 447–447. Springer Berlin Heidelberg, 1986.
13. A. Poschmann, S. Ling, and H. Wang. 256 bit standardized crypto for 650 ge: Gost revisited. In *Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*, CHES'10, pages 219–233, Berlin, Heidelberg, 2010. Springer-Verlag.