Three-Subset Meet-in-the-Middle Attack on Reduced XTEA

Abstract. This paper presents an improved single-key attack on a block-cipher XTEA by using the three-subset meet-in-the-middle (MitM) attack. Firstly, a technique on a generic block-ciphers is discussed. It points out that the previous work applying the splice-and-cut technique to the three-subset MitM attack contains incomplete arguments, and thus it requires a very large data complexity, which is close to the code book. This paper gives a corrected procedure to keep the data complexity small. Secondly, the three-subset MitM attack is applied for reduced-round XTEA, which is a 64-bit block-cipher with 64-round Feistel network and a 128-bit key. 25 rounds are attacked with 9 known plaintexts and $2^{120.40}$ XTEA computations, while the previous best single-key attack only reaches 23 rounds. In the chosen-plaintext model, the attack is extended to 28 rounds with 2^{37} chosen-plaintexts and $2^{120.38}$ computations. Finally, the number of attacked rounds is optimized by allowing a marginal improvement of the complexity, which will be useful to understand the limit of the attack. It is shown that the MitM attack can potentially work up to 31 rounds.

keywords: XTEA, 3-subset meet-in-the-middle, splice-and-cut, multi-pair match

1 Introduction

Block-ciphers are one of the most fundamental symmetric-key primitives. In recent years, block-ciphers are required to be implemented in the resource-restricted environment such as an RFID tag. Due to this, the design and security of light-weight block-ciphers are receiving great attentions.

At FSE 1994, the block-cipher TEA was proposed by Wheeler and Needham [29]. Then, a series of block-ciphers called the TEA-family have been proposed.

– TEA was adopted in the Microsoft's Xbox gaming console for providing the hash function facility. However, an attack was proposed by Steil [26], which was based on a pioneering work by Kelsey, Schneier, and Wagner [14]. They also proposed a related-key attack [15].
– As a stronger version of TEA, Needham and Wheeler designed XTEA and Block TEA [20]. TEA and XTEA are implemented in the Linux kernel.
– To correct the weaknesses of Block TEA, Needham and Wheeler designed Corrected Block TEA, which is called XXTEA [21]. However, an attack on full Block TEA and several weaknesses of XXTEA were pointed out by Saarinen [22]. Full XXTEA was also attacked by Yarrkov [30].

The full XTEA has not been attacked yet. In addition, XTEA is known to be suitable for the resource-restricted environment [13]. Hence, investigating the security of XTEA seems interesting and useful.

Several attacks on round-reduced XTEA are known. Related-key attacks and weak-key attacks can work on many rounds of XTEA [7, 16–18]. On the other hand, single-key attacks without the weak-key assumption are more important in terms of the impact in practice. After the work by Moon *et al.* in 2002 [19] and Hong *et al.* in 2003 [11], the research on the single-key attack was stopped for a while. In 2011, Sekar *et al.* presented an attack on 23 rounds out of 64 rounds of XTEA with the MitM attack [25]. Recently, Chen *et al.* have presented another attack on 23 rounds with the impossible differential approach [8].

The MitM attack was introduced by Diffie and Hellman in 1997 [10]. The basic idea is separating the block-cipher E_K into two sub-parts $E1_{k1}$ and $E2_{k2}$, where $k1$ and $k2$ are independent subkeys and $E_K = E2_{k2} \circ E1_{k1}$. For a pair of plaintext P and ciphertext C, an attacker independently computes $E1_{k1}(P)$ and $E2^{-1}_{k2}(C)$ for all possible $k1$ and $k2$, respectively. By checking the match of two results, wrong key candidates can be eliminated efficiently. The MitM attack is now widely applied to various block-ciphers. Note that several attacks that are a little bit different from the basic MitM attack are also called MitM attack. The attacks on AES [9] and XTEA [25] are such examples.

In recent years, the MitM attack was applied to various hash functions e.g. MD5 [3, 24], SHA-1 [2], SHA-2 [1], and many techniques were developed for the MitM attack on hash functions. In 2010, Bogdanov and Rechberger revisited the MitM attack on block-ciphers and proposed a framework called three-subset MitM attack [6]. It works efficiently against block-ciphers with a weak key schedule. This matches the property of light-weight ciphers on which a heavy key schedule cannot be implemented. Hence, the three-subset MitM attack is actively discussed presently. In addition, several researchers rearranged the techniques which were originally developed for the hash functions, and applied them to block-ciphers [4, 5, 12, 27, 28].

Our Contributions

In this paper, we firstly point out the incomplete argument of the previous work [27], which applies the splice-and-cut technique [3] for the three-subset MitM attack. We show that its simple application will lead to a very large data complexity. We then propose a corrected procedure to keep the data complexity small. This procedure can be used for any block-cipher in generic.

Secondly, by using this technique, the three-subset MitM attack is applied for reduced-round XTEA. With some non-negligible improvement of the complexity, the attack reaches 25 rounds in the known-plaintext model and 28 rounds in the chosen-plaintext model. The complexity of the attack is summarized in Table 1. As far as we know, our attack is the best in terms of the number of attacked rounds in a single-key setting. Finally, to discuss the further extension of the attack and to understand the limit of the attack, we optimize the number of attacked rounds by allowing a marginal improvement of the complexity. We show that the three-subset MitM attack potentially works up to 29 and 31 rounds in the known-plaintext and chosen-plaintext models, respectively.

Table 1. Comparison of key recovery attacks on reduced XTEA

Single-key Attacks					Related-key Attacks				
Approach	Rounds	Time	Data	Ref.	Weak-key	Rounds	Time	Data	Ref.
Imp. Diff.	14	2^{85}	$2^{62.5}$ CPs	[19]		27	$2^{115.15}$	$2^{20.5}$ CPs	[16]
Trunc. Diff.	23	$2^{120.65}$	$2^{20.55}$ CPs	[11]	Yes	34	$2^{31.94}$	2^{62} CPs	[17]
MitM	23	$2^{117.00}$	18 KPs	[25]		36	$2^{126.44}$	$2^{64.98}$ CPs	[18]
Imp. Diff.	23	$2^{116.9}$	2^{62} CPs	[8]	Yes	36	$2^{104.33}$	$2^{63.83}$ CPs	[18]
3-Sub. MitM	25	$2^{120.40}$	9 KPs	**Ours**		37	2^{125}	2^{63} CPs	[7]
3-Sub. MitM	28	$2^{120.38}$	2^{37} CPs	**Ours**	Yes	51	2^{123}	2^{63} CPs	[7]
3-Sub. MitM	29	$2^{127.60}$	2 KPs	**Ours**					
3-Sub. MitM	31	$2^{127.54}$	6 CPs	**Ours**					

Paper Outline

Section 2 describes the specification of XTEA. Section 3 summarizes the previous work. Section 4 explains the incompleteness of the previous attack procedure and describes a new generic technique for the three subset MitM attack on block-ciphers. Section 5 shows attacks on XTEA. Section 6 shows attacks on XTEA with a marginal improvement of the complexity. Finally, we conclude the paper in Sect. 7.

2 Specification of XTEA

XTEA [20] has the block size of 64 bits and key size of 128 bits. It uses a 64-round Feistel network. The F-function of the Feistel network takes a 32-bit input x and produces a 32-bit output as:

$$F(x) = ((x \ll 4) \oplus (x \gg 5)) \boxplus x, \tag{1}$$

where '$\ll$' and '$\gg$' represent the left and right shift (not rotation) respectively, '$\oplus$' represents the XOR, and '$\boxplus$' represents the modular addition over 2^{32}. The round function is described in Fig. 1

The 128-bit key K is separated into four 32-bit subkeys K_0, K_1, K_2, and K_3. The 64-bit input to round i consists of two 32-bit values L_{i-1} and R_{i-1}. First, the plaintext P is loaded into the state, namely $L_0 \| R_0 \leftarrow P$. Then, the following are computed recursively for $i = 0, 1, \ldots, 63$;

$$L_{i+1} \leftarrow R_i, \tag{2}$$

$$R_{i+1} \leftarrow L_i \boxplus ((\delta_i \boxplus K_{\pi(i)}) \oplus F(R_i)), \tag{3}$$

where δ_i is a pre-defined constant and $\pi(i) \in \{0, 1, 2, 3\}$ is a pre-defined subkey index in round i. The list of index $\pi(i)$ is shown in Table 2. Finally, $L_{64} \| R_{64}$ are produced as the ciphertext of P.

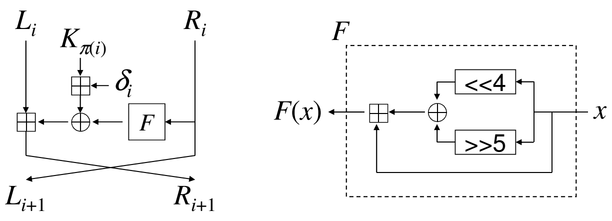

Fig. 1. Round function of XTEA

Table 2. Subkey index for each round

Round i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Index $\pi(i)$	0	3	1	2	2	1	3	0	0	0	1	3	2	2	3	1
Round i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Index $\pi(i)$	0	0	1	0	2	3	3	2	0	1	1	1	2	0	3	3
Round i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Index $\pi(i)$	0	2	1	1	2	1	3	0	0	3	1	2	2	1	3	1
Round i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Index $\pi(i)$	0	0	1	3	2	2	3	2	0	1	1	0	2	3	3	2

3 Previous Work

3.1 Previous Meet-in-the-Middle Attack on XTEA

At CT-RSA2011, Sekar *et al.* presented a meet-in-the-middle attack on several intermediate 23 rounds of XTEA [25], e.g. from round 16 to 38, in total 23 rounds. Let us denote the data processing transformation for 23 rounds by E_{23}. The main idea of this attack is separating E_{23} into an inner round I and outer rounds O_1 and O_2, namely $E_{23}(\cdot) = O_2 \circ I \circ O_1(\cdot)$, where one of the subkeys never appears in the outer rounds. Because all subkeys in the outer rounds are also used in an inner round, it is impossible to compute the inner and outer rounds independently, and thus this attack is a little bit different from the classical meet-in-the-middle attack.

3.2 Three-Subset Meet-in-the-Middle Attack

The three-subset meet-in-the-middle attack is an approach of cryptanalysis on block-ciphers proposed by Bogdanov and Rechberger at SAC2010 [6]. The framework is well summarized in [12] by Isobe. This attack essentially examines all possible key candidates. Because wrong keys can be filtered out efficiently by using the meet-in-the-middle techniques, the attack can be faster than the exhaustive search.

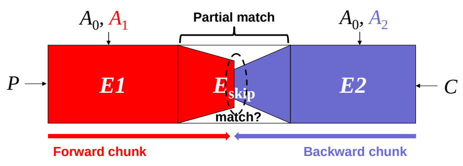

Fig. 2. Basic framework of 3-subset MitM attack with the partial match

Basic framework. A block-cipher E_K with the block size of b bits is divided into two parts $E1_{K1}$ and $E2_{K2}$ so that $E_K(x) = E2_{K2} \circ E1_{K1}(x), x \in \{0,1\}^b$, where $K1$ and $K2$ are sets of key bits involved in the computations of $E1$ and $E2$, respectively. $E1_{K1}$ is called the *forward chunk* and $E2_{K2}$ is called the *backward chunk*. $A_0 = K1 \cap K2$ is a common set of key bits used in both $E1$ and $E2$. $A_1 = K1 \setminus K1 \cap K2$ and $A_2 = K2 \setminus K1 \cap K2$ are sets of key bits used in only $E1$ and only $E2$, respectively. The attack first prepares a pair of plaintext and ciphertext denoted by (P, C). The attack procedure is as follows.

MitM phase
1. For all candidates of A_0, do as follows;
2. Compute $E1_{K1}(P)$ for all candidates of A_1, and store them in a table T.
3. For each value in A_2, compute $E2^{-1}_{K2}(C)$, and check whether or not the same value is in T. If the value exists, the key is stored as a valid key candidate.

Exhaustive search phase
4. Obtain another pair of plaintext and ciphertext denoted by (P', C'). For all valid key candidates, exhaustively check whether $E_K(P') = C'$ holds. If the equation holds, the key is stored as a valid key candidate.
5. Repeat Step 4 until the key space is reduced to 1.

Let the size of the secret key be ℓ. After the MitM phase, the key candidate space is reduced to $2^{\ell-b}$. With each iteration of the exhaustive search phase, the key candidate space is reduced by a factor of b bits. In step 2, $2^{|A_1|}$ values are stored, where $|A_1|$ is the number of elements in A_1. Therefore, the attack requires a memory to store $2^{|A_1|}$ values. The computational complexity for the MitM phase is $2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})$ and the computational complexity for the exhaustive search phase is $2^{\ell-b} + 2^{\ell-2b} + \cdots$. Most of the case, the dominant complexity is $2^{|A_0|}(2^{|A_1|} + 2^{|A_2|})$. When $|A_1| = |A_2|$, the attack can be faster than the brute force attack by a factor of $2^{|A_1|}$. Therefore, finding two chunks with large $|A_1|$ and $|A_2|$ is important for the attacker.

Note that, instead of separating E_K into $E1_{K1}$ and $E2_{K2}$, it is often separated into three parts so that $E_K(\cdot) = E2_{K2} \circ E_{\text{skip}} \circ E1_{K1}(\cdot)$. This is illustrated in Fig. 2. In this case, the direct comparison of $E1_{K1}(P)$ and $E2^{-1}_{K2}(C)$ becomes impossible due to the existence of E_{skip}. However, the part of the internal state inside E_{skip} can often be computed without the knowledge of subkeys. This is called *partial match*. As a result, by reducing the number of matched bits, the number of attacked rounds can be increased. Let m be the number of matched

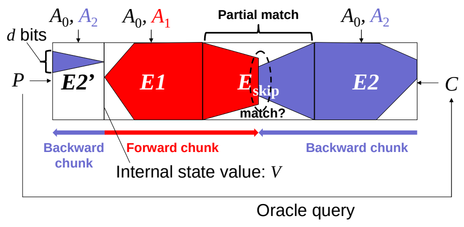

Fig. 3. Splice-and-cut technique for 3-subset MitM attack

bits. As long as $m \geq |A_1| + |A_2|$, the attack is faster than the brute force attack by a factor of $2^{|A_1|}$. However, if $m < |A_1| + |A_2|$, the improved factor is reduced to 2^m. Hence, keeping m big is also important to mount an efficient MitM attack.

Recent progress. Before Bogdanov and Rechberger presented the 3-subset meet-in-the-middle attack, the meet-in-the-middle attack was applied to various hash functions, e.g. MD5 [3, 24], HAVAL [23], SHA-1 [2], and SHA-2 [1]. Through these attacks, a lot of techniques for the meet-in-the-middle attack were proposed such as *splice-and-cut* [3] and *initial-structure* [24]. These techniques were developed in order to attack hash functions, and thus they cannot be directly applied to the 3-subset meet-in-the-middle attack which targets block-ciphers. However, several researchers reconstructed the techniques and successfully applied them to block-ciphers.

Wei *et al.* [27] applied the splice-and-cut technique [3] to the block-cipher KTANTAN. This is illustrated in Fig. 3. This enables the attacker to divide the target cipher E_K into $E2_{K2} \circ E_{\text{skip}} \circ E1_{K1} \circ E2'_{K2}$ instead of $E2_{K2} \circ E_{\text{skip}} \circ E1_{K1}$. Obviously, the search space of the chunk separation increases. The attack first fixes the value of the internal state between $E1_{K1}$ and $E2'_{K2}$, which is denoted by V. In the backward chunk, the attacker computes $P = E2'^{-1}_{K2}(V)$ for all possible values of A_2, and query P to the encryption oracle to obtain the corresponding ciphertext C. Then, $E2^{-1}_{K2}(C)$ is computed. Assume that the change of A_2 in the backward chunk only propagates to the partial bits (say d bits) in P. If the attacker has 2^d data (chosen plaintexts) for all possible such P in advance, the corresponding C can be queried. The data complexity increases depending on d, namely how the change of A_2 propagates to P. As [1] noted, we should avoid that the change of A_2 propagates to all bits in P. Otherwise, the attack requires the full code book. However, in Sect. 4, we will show that the discussion in [27] is not enough to perform the attack only with 2^d chosen plaintexts.

Bogdanov *et al.* [5] proposed a useful form of the initial-structure and applied it to full-round AES-128. The technique is called *biclique*. Because we cannot find the application of the biclique for XTEA, we do not discuss details of the biclique.

4 Maintaining Small Data Complexity in Three-Subset MitM Attack with Splice-and-Cut Technique

In this section, we show that the discussion in [27] is not enough to perform the attack only with 2^d chosen plaintexts. We then propose the corrected procedure to keep the small data complexity even if the splice-and-cut technique is applied.

Let us use Fig. 3 to discuss the problem. The estimation of the data complexity by [27] can be summarized as follows.

> *Suppose that changing the value of A_2 only gives impact to d-bits of the plaintext after computing $P = E2'^{-1}_{K2}(V)$. Then, the attack can be performed with 2^d chosen plaintexts.*

This argument is surely true as long as the value of A_0 is fixed. However, the meet-in-the-middle attack is repeated many times with changing the value of A_0, and the simple application of the procedure requires the data complexity of $|A_0| \times 2^d$. In many block-ciphers including KTANTAN and XTEA, the key size is bigger than the block size b. This means that the value of $|A_0|$ will be much bigger than the block size, and thus the data complexity becomes the same as the entire code book. Wei *et al.* [27] did not discuss the case where A_0 is changed.

The solution of this problem is that every time we change the value of A_0, we also change the internal state value V so that $b - d$ bits in the plaintext, which are not affected by A_2, always take the same value. However, achieving this is sometimes hard especially if the target cipher uses the Feistel network.

Let us discuss the case that $E2'^{-1}_{K2}$ covers 2 rounds of the Feistel ciphers whose round function consists of the key addition and a certain transformation F. This is depicted in Fig. 4. We assume that only a part of bits in $K_{\pi(i)}$ belong to A_2 and the other bits of $K_{\pi(i)}$ and all bits of $K_{\pi(i-1)}$ belong to A_0.

The impacts of changing A_2 is marked with blue lines, and we assume that only a limited number of bits in blue lines are influenced by changing A_2. We now consider the impact of changing the value of A_0. If we change the bits of $K_{\pi(i)}$ belonging to A_0 into various values, the value of $F(L_{i+1} \oplus K_{\pi(i)})$ takes various values. If the value of L_{i+1} is fixed in advance, we can compute $F(L_{i+1} \oplus K_{\pi(i)})$ and thus can modify R_{i+1} so that $b - d$ bits of R_{i-1} are always the fixed value. Then, in round $i - 1$, we also need to absorb the change of $F(L_i \oplus K_{\pi(i-1)})$ by modifying R_i. However, this is the contradiction in round i, where the value of $L_{i+1}(= R_i)$ needs to be fixed in advance. Consequently, fixing the $b - d$ bits of the plaintext for any A_0 cannot be done trivially.

We finally show that this problem can be solved, that is to say, the attack can work only with 2^d chosen plaintexts regardless of the cipher's structure. The procedure is depicted in Fig. 5. Assume that during the computation in $E2'^{-1}_{K2}$, changing bits in A_2 only gives influence to d bits of the plaintext P. Without losing the generality, we assume that the left d bits are influenced. We denote the left d bits of P by $P_{0,d-1}$, and thus P is denoted by $P_{0,d-1} \| P_{d,b-1}$. At the first, we choose a unique value for $b - d$ bits of $P_{d,b-1}$. We denote this value by C_{PT}. The goal is choosing the internal state value V so that C_{PT} can be always achieved for A_0.

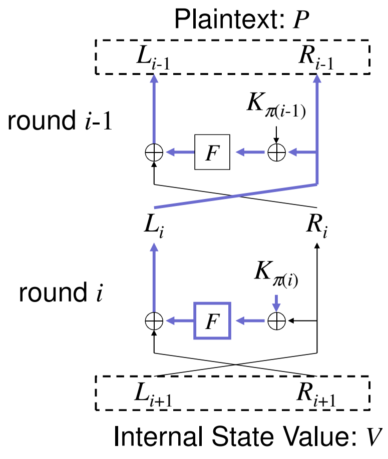

Fig. 4. A problem of the splice-and-cut technique for block-ciphers. First, $L_{i+1}(= R_i)$ is fixed. Second, R_{i+1} is modified in round i. Third, R_i is modified in round $i-1$, but this is the contradiction against the first step.

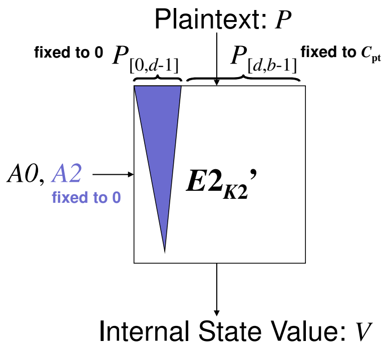

Fig. 5. Computation of V which avoids increasing the data complexity

1. Every time the bits in A_0 are chosen, do as follows;
2. Temporarily fix the value of d bits of $P_{0,d-1}$ and all bits in A_2, say 0.
3. Compute $V = E2'_{K2}(P_{0,d-1} \| P_{d,b-1})$ with chosen A_0 and fixed A_2.

When you perform the meet-in-the-middle attack with the configuration of chosen A_0, use V as the internal state value. Then, $P_{d,b-1} = C_{\text{PT}}$ is always satisfied, and the attack thus can be performed with only 2^d chosen plaintexts.

5 Attacks on XTEA

5.1 Known Plaintext Attack on 25-Round XTEA

In this section, we propose a known plaintext attack on 25 rounds of XTEA with the single-key setting. Although the attack cost is a little bit worse than the previous best attack [25], the number of attacked rounds is extended. Note that, in the known plaintext attack, the splice-and-cut technique cannot be used.

Chunk separation. The chunk separation is shown in Table 3. The attacked rounds are from round 5 to 29, in total 25 rounds. The forward chunk covers the first 7 rounds, in which K_2 is not used in these rounds. The backward chunk covers the last 7 rounds, in which K_3 is not used in these rounds. Rounds 12 to 22, in total 11 rounds are skipped. In order to perform the match over skipped 11 rounds, we fix the least significant 21 bits of K_3 and K_2, and use the

Table 3. Chunk separation for known plaintext attack on 25-round XTEA

Round i	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Index $\pi(i)$	1	3	0	0	0	1	3	2	2	3	1	0	0	1	0	2	3	3	2	0	1	1	1	2	0
	forward chunk							E_{skip}											backward chunk						

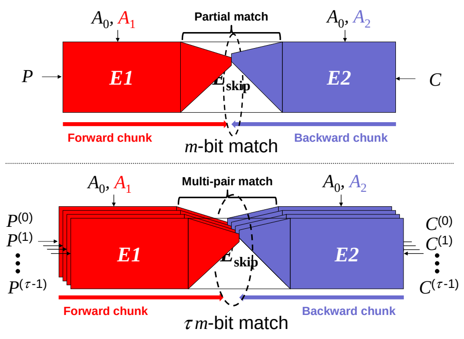

Fig. 6. (Top) match with one pair, (Bottom) multi-pair match

most significant 11 bits as the free bits for each chunk. Namely, $|A_1| = |A_2| = 11$. Inside E_{skip}, we only check the match of 1 bit for a pair of plaintext and ciphertext, namely $m = 1$. As explained in Sect. 3.2, because $m < |A_1| = |A_2|$, the simple application of the basic framework improves the complexity only by a factor of 2^1. In order to improve the attack efficiency, we use multi-pairs of plaintext/ciphertext $(P^{(j)}, C^{(j)})$ to check the match of two chunks.

Multi-pair match. Assume $m = 1$ like our attack. Therefore, each plaintext/ciphertext pair produces 1-bit filtering condition. To filter out wrong key candidates more efficiently, we use multi-pairs rather than a single pair. This is illustrated in Fig. 6. As a result, with τ plaintext/ciphertext pairs, the 1-bit filtering condition is amplified into a τ-bit filtering function. The time complexity for each chunk becomes τ times (linear increase) to analyze τ pairs, instead 2^τ wrong key candidates can be filtered out (exponential increase). Hence, the efficiency of the entire attack increases.

In the following, we explain how to compute 1 matching bit during skipped 11 rounds in our attack. In the forward chunk, until the output of round 11 (L_{12}, R_{12}) can be computed in all bits. For round 12, in the computation of

$$R_{13} = L_{12} \boxplus ((\delta_{12} \boxplus K_2) \oplus F(R_{12})), \tag{4}$$

we do not know the value of the bit positions 21 to 31 of K_2, but we know the bit positions 0 to 20 of K_2. Hence, we cannot compute the bit positions 21 to 31 of R_{13} but can compute the bit positions 0 to 20 of R_{13}.

Table 4. Partial-match for 25-round XTEA. Each entry shows the known bit positions.

Forward chunk			Backward chunk		
j	L_j	R_j	j	L_j	R_j
12	31–0	31–0	23	31–0	31–0
13	31–0	20–0	22	20–0	31–0
14	20–0	15–0	21	15–0	20–0
15	15–0	10–0	20	10–0	15–0
16	10–0	5–0	19	5–0	10–0
17	5–0	0	18	0	5–0
18	0	?			

For round 13, R_{14} is computed as follows.

$$R_{14} = L_{13} \boxplus ((\delta_{13} \boxplus K_2) \oplus F(R_{13})), \tag{5}$$

$$F(R_{13}) = ((R_{13} \ll 4) \oplus (R_{13} \gg 5)) \boxplus R_{13}. \tag{6}$$

Because only the bit positions 0 to 20 of R_{13} are known, we can only compute the bit positions 0 to 15 of $F(R_{13})$. Hence, we can compute the bit positions 0 to 15 of R_{14}.

Similarly, we can trace the bit positions that can be computed. The result is shown in Table 4. As we process one round, the number of known bits decreases by 5 bits. Hence, we can compute the bit position 0 of L_{18}. The result for the backward chunk can be obtained in the same way, which is also shown in Table 4. We can compute the bit position 0 of L_{18}. Finally, we can check the match for 1 bit (bit position 0) of L_{18}.

To perform the multi-pair match, we use $\tau = 9$ pairs of known plaintext/ciphertext and apply the same matching method for 9 pairs simultaneously. Hence, we can check the match of 9 bits in total.

Attack procedure. At first, the attacker prepares $\tau = 9$ pairs of known plaintexts and ciphertexts denoted by $(P^{(0)}, C^{(0)}), (P^{(0)}, C^{(0)}), \ldots, (P^{8(0)}, C^{(8)})$.

1. Guess the value of A_0, namely the value of K_0, K_1, bit positions 0–20 of K_2, and bit positions 0–20 of K_3.
2. For all possible candidates of A_1, namely for the bit positions 21 to 31 in total 11 bits of K_2, compute the round function from round 5 to 11 for $P^{(j)}$, where $0 \leq j \leq 8$, and store the value of $(L_{12}^{(j)}, R_{12}^{(j)})$ and corresponding K_2. Then partially compute the round function from round 12 to 17 and store the value of $L_{18}^{(j)}$ in bit position 0.
3. For all possible candidates of A_2, namely for the bit positions 21 to 31 in total 11 bits of K_3, compute the round function from round 29 to 23 for $C^{(j)}$, $0 \leq j \leq 8$, and store the value of $(L_{23}^{(j)}, R_{23}^{(j)})$ and corresponding K_3. Then partially compute the round function from round 22 to 18 and check whether or not the value of $L_{18}^{(j)}$ in bit position 0 matches with the value from the forward chunk.

4. If they match for all j, compute the round function from round 12 to 17 and 23 to 18 in all bits for the pair $(P^{(0)}, C^{(0)})$, and check whether or not the remaining 63 bits of $(L_{18}^{(0)}, R_{18}^{(0)})$ match.
5. If all bits match, examine $E_K(P^{(j)})$ equals to $C^{(j)}$ for all j. If they match for all j, the key candidate is the correct key. Otherwise, the key is wrong, and repeat the attack from Step 1 with a different guess of A_0.

Complexity evaluation. Let us assume that 1 round operation is $\frac{1}{25}$ XTEA computation. Step 2 requires $9 \times 2^{11} \times \frac{12}{25}$ XTEA computations and Step 3 requires $9 \times 2^{11} \times \frac{12}{25}$ XTEA computations. After the 9-bit match at Step 3, $2^{11+11-9} = 2^{13}$ pairs will survive. Step 4 requires $2^{13} \times \frac{12}{25}$ XTEA computations. The complexity for Step 5 is negligible because the number of right key candidates are already reduced at Step 4. Finally, these procedures are iterated for $2^{|A_0|} = 2^{128-11-11} = 2^{106}$ times due to the recursion at Step 1. Hence, the total computational cost is

$$2^{106}(9 \times 2^{11} \times \frac{12}{25} + 9 \times 2^{11} \times \frac{12}{25} + 2^{13} \times \frac{12}{25}), \tag{7}$$

which is approximately $2^{120.40}$ XTEA computations. The attack requires to store $(2^{11} \times 4)$ 32-bit values at Step 2 and Step 3. Hence, the total required memory is 2^{14} 32-bit values, which correspond to 2^{13} XTEA state.

Remarks for the choice of τ. The number of known plaintext/ciphertext pairs τ is chosen to optimize the time complexity. With τ pairs, the time complexity of the attack in Eq. (7) can be written as follows.

$$\begin{aligned}
\text{Time} &= 2^{106}(\tau \times 2^{11} \times \frac{12}{25} + \tau \times 2^{11} \times \frac{12}{25} + 2^{22-\tau} \times \frac{12}{25}), && (8)\\
&= \frac{12}{25} \times 2^{106}(\tau \times 2^{12} + 2^{22-\tau}), && (9)\\
&= \frac{12}{25} \times 2^{118}(\tau + 2^{10-\tau}). && (10)
\end{aligned}$$

When τ is an integer, $\tau = 9$ or 10 minimizes the equation. Hence, we chose $\tau = 9$. As is indicated above, a trade-off exists in this attack. The attack, with some increase of the time complexity, still can work even if only τ pairs where $\tau < 9$ is available.

5.2 Chosen Plaintext Attack on 28-Round XTEA

In the chosen plaintext scenario, the attack can be extended up to 28 rounds. We add three rounds to the beginning of the 25-round attack in Sect. 5.1 by using the splice-and-cut technique.

Table 5. Chunk separation for chosen plaintext attack on 28-round XTEA

Round i	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Index $\pi(i)$	1	2	2	1	3	0	0	0	1	3	2	2	3	1	0	0	1	0	2	3	3	2	0	1	1	1	2	0
	backward chunk			forward chunk							E_{skip}											backward chunk						

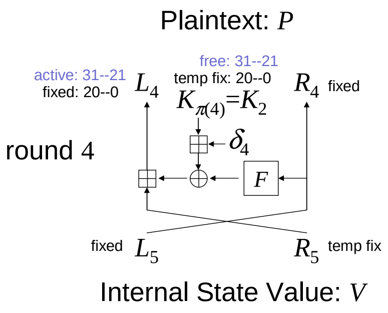

Fig. 7. Splice-and-cut for 1 round

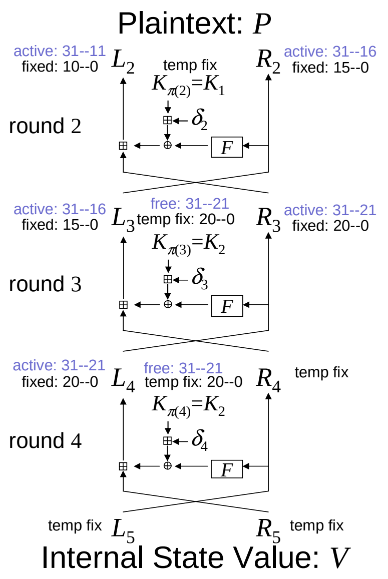

Fig. 8. Splice-and-cut for 3 rounds

Chunk separation. The chunk separation is shown in Table 5. The attacked rounds are from round 2 to 29, in total 28 rounds. Compared to Table 3, 3 rounds are added as a part of backward chunk before round 5. In fact, K_3 is not used in these rounds. Therefore, the attack is the same as 25-round known-key attack except for the application of the splice-and-cut technique. In the following part, we explain how the splice-and-cut technique work in this attack.

1-round splice-and-cut. For simplicity, we firstly explain the splice-and-cut technique for 1-round, namely the attack on 26 rounds (from rounds 4 to 29).

The computation in the backward chunk in round 4 is depicted in Fig. 7. *fixed* represents the values which are fixed throughout the attack. *temporarily fixed* represents the values which are fixed during the meet-in-the-middle phase but are changed to repeat the meet-in-the-middle phase. *free* represents the free bits in the meet-in-the-middle phase. *active* represents the values which are influenced by free bits. The free bits are bit positions 21 to 31 of K_2. Changing the free bits will impact to the bit positions 21 to 31 of the left-half of the plaintext (L_4). Therefore, in this attack, we firstly prepare 2^{11} chosen plaintexts where the bit positions 0 to 20 of L_4 and all bits of R_4 are fixed to a pre-chosen constant (any value is acceptable, say 0) and the bit positions 21 to 31 of L_4 take all possibilities. In the end, the data complexity is 2^{11} chosen plaintexts.

As pointed out in Sect. 4, it must be ensured that for any value of the bit positions 0 to 20 of K_2, we choose appropriate value of L_5 so that the fixed value (the bit positions 0 to 20) of L_4 can be realized. This is possible by using the procedure in Sect. 4. In details, we set the values labelled as *fixed* to pre-chosen values, and fix the values labelled as *active* in L_4 and *free* in K_2 to any value, say 0. Because L_4, R_4, and K_2 are fixed now, we can compute temporarily fixed values L_5 and R_5 by simply computing the round function. The resulting V ensures that for any value of free bits in K_2, fixed values in the plaintext (L_4 and R_4) are always achieved.

Combination of the splice-and-cut and multi-pair match. These two techniques can be used at the same time. Let us firstly consider the simple method. When the interstate value V is computed, we prepare $\tau = 9$ patterns, say 0 to 8, for the *fixed values* of the plaintext, and compute the corresponding 9 patterns of V. This method requires 2^{11} chosen plaintexts for each pattern of the *fixed values* of the plaintext, thus $\tau \cdot 2^d = 9 \cdot 2^{11}$ chosen plaintexts in total.

By carefully choosing V, we can reduce the data complexity to $2^d = 2^{11}$. The idea is using the bit positions in V which the change of their bits only gives influence to the active bits of the plaintext. In the case of Fig. 7, the bit positions 21 to 31 of R_5 only gives influence to the active bits in the plaintext. In the end, the procedure to obtain τ intermediate state Vs is as follows.

1. Obtain one $V^{(0)}$ with the same manner as the case for the single pair.
2. Modify V^0 in bit positions discussed above to $\tau - 1$ patterns to obtain $V^{(1)}, V^{(2)}, \ldots, V^{(\tau-1)}$.

These τ intermediate values ensure that for any free bits in K_2, the fixed bits in L_4 can be achieved. Hence, the multi-pair match can be performed only with 2^d chosen plaintexts.

3-round splice-and-cut. We show the 3-round backward computations for the attack on 28 rounds. See Fig. 8 for its illustration. After 2-round computations, the number of influenced bits is 27 and after 3-rounds, it becomes 37. In the end, the 3-rounds are appended with the splice-and-cut technique by using 2^{37} chosen plain texts.

Brief description of the attack.

1. Prepare 2^{37} chosen plaintexts and obtain the corresponding ciphertexts.
2. For each candidate of A_0, do as follows.
3. Compute $V^{(j)}$, where $0 \leq j \leq 8$.
4. For all candidates of A_1, compute the forward chunk starting from $V^{(j)}$.

Table 6. Chunk separation for known plaintext attack on 29-round XTEA

Round i	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
Index $\pi(i)$	2	0	1	1	1	2	0	3	3	0	2	1	1	2	1	3	0	0	3	1	2	2	1	3	1	0	0	1	3
	forward chunk							E_{skip}															backward chunk						

5. For all candidates of A_2, compute rounds 4 to 2 starting from $V^{(j)}$ to obtain $P^{(j)}$. By using the oracle query, obtain the corresponding ciphertext $C^{(j)}$ and compute the backward chunk.

The rest of the procedure is exactly the same as 25-round known-key attack. The computational cost slightly decreases because the ratio of the independently computed part becomes bigger, which is $2^{120.38}$ XTEA computations.

6 Discussion toward the Further Extension

In this section, we extend the number of attacked rounds as much as possible by allowing a marginal improvement from the exhaustive search. One may say that these attacks are useless because their complexities may be worse than the optimization of the brute force attack, which all keys are examined but many computations are shared for different keys and thus the computational cost for each key is less than the straight-forward brute force attack. This statement may be true. However, these attacks follow the standard framework of the MitM attack with parameters $|A_1| = |A_2| = m = 1$, and filter out wrong-keys in a clever way. The optimized brute force attack can be somehow applied for almost all block-ciphers in practice, while the applicability of the MitM attack heavily depends on the cipher's structure. Therefore, we believe that these attacks will help to advance the state-of-the-art about the security evaluation of XTEA.

6.1 Known Plaintext Attack on 29-Round XTEA

Chunk separation. The chunk separation is shown in Table 6. The attacked rounds are from round 23 to 51, in total 29 rounds. The forward chunk covers the first 7 rounds, in which K_3 is not used in these rounds. The backward chunk covers the last 7 rounds, in which K_2 is not used in these rounds. Rounds 30 to 44, in total 15 rounds are skipped. In order to perform the match over skipped 15 rounds, we fix the least significant 31 bits of K_3 and K_2, and only use the most significant 1 bit as the free bit for each chunk. Namely, $|A_1| = |A_2| = 1$.

Partial-match. The partial-match procedure is almost the same as Sect. 5.1. To avoid the redundancy, we only show the result in Table 7. As we process one round, the number of known bits decreases by 5 bits. Hence, we can compute the bit position 0 of L_{38}. The result for the backward chunk can be obtained in the same way, which is also shown in Table 7. We can compute the bit position 0 of L_{38}. Finally, we can compare the match of 1 bit (bit position 0) of L_{38}.

Table 7. Partial-match for 29-rounds. Each entry shows the known bit positions.

Forward chunk			Backward chunk		
j	L_j	R_j	j	L_j	R_j
30	31–0	31–0	45	31–0	31–0
31	31–0	30–0	44	30–0	31–0
32	30–0	25–0	43	25–0	30–0
33	25–0	20–0	42	20–0	25–0
34	20–0	15–0	41	15–0	20–0
35	15–0	10–0	40	10–0	15–0
36	10–0	5–0	39	5–0	10–0
37	5–0	0	38	0	5–0
38	0	?			

Table 8. Bit positions independent of the free bits of the backward chunk

Plaintext P		
j	L_j	R_j
21	25–0	30–0
22	30–0	31–0
23	31–0	31–0
Internal state V		

Attack procedure. At first, the attacker prepares 2 pairs of known plaintexts and ciphertexts $(P^{(0)}, C^{(0)})$ and $(P^{(1)}, C^{(1)})$.

1. Guess the value of A_0, namely the value of K_0, K_1, bit positions 0–30 of K_2, and bit positions 0–30 of K_3.
2. For all possible candidates of A_1, namely for 2 choices of the bit position 31 of K_2, compute the round function from round 23 to 29 for $P^{(0)}$, and store (L_{30}, R_{30}) and corresponding K_2. Then partially compute the round function from round 30 to 37 and store L_{38} in bit position 0.
3. For all possible candidates of A_2, namely for 2 choices of the bit position 31 of K_3, compute the round function from round 51 to 45 for $C^{(0)}$, and store (L_{45}, R_{45}) and corresponding K_3. Then partially compute the round function from round 44 to 38 and check whether or not the value of L_{38} in bit position 0 matches the value from the forward chunk.
4. If they match, compute the round function from round 30 to 37 and 44 to 38 in all bits, and check if the remaining 63 bits of (L_{38}, R_{38}) match.
5. If all bits match, check whether or not $E_K(P^{(1)})$ equals to $C^{(1)}$. If they match, the key candidate is the correct key. Otherwise, the key is wrong, and repeat the attack from Step 1 with a different guess of A_0.

Complexity evaluation. Step 2 requires $2 \times \frac{15}{29}$ XTEA computations and Step 3 requires $2 \times \frac{14}{29}$ XTEA computations. After the 1 bit match at Step 3, $2^{1+1-1} = 2^1$ pairs will survive. Step 4 requires $2^1 \times \frac{15}{29}$ XTEA computations. The complexity for Step 5 is negligible. Finally, these procedures are iterated for $2^{|A_0|} = 2^{128-1-1} = 2^{126}$ times. Hence, the total computational cost is $2^{126}(2 \times \frac{15}{29} + 2 \times \frac{14}{29} + 2 \times \frac{15}{29})$, which is $2^{127} \times \frac{44}{29} \approx 2^{127.60}$ XTEA computations. The attack requires to store (2×4) 32-bit values at Step 2 and Step 3. Hence, the total required memory is 2^4 32-bit values, which correspond to 2^3 XTEA state.

Table 9. Chunk separation for known plaintext attack on 31-round XTEA

i	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
$\pi(i)$	3	3	2	0	1	1	1	2	0	3	3	0	2	1	1	2	1	3	0	0	3	1	2	2	1	3	1	0	0	1	3
backward			forward chunk							E_{skip}															backward chunk						

6.2 Chosen Plaintext Attack on 31-Round XTEA

Chunk separation. The chunk separation is shown in Table 9. It adds two rounds to the chunk for the known plaintext attack on 29 rounds. We only use the most significant 1 bit as the free bit for each chunk. Note that you can add two rounds to the end of the 29-round attack instead of the beginning. In that case, the attack becomes the chosen ciphertext attack.

Difference from the known plaintext attack. The data complexity depends on the number of bits in the plaintext that are influenced by the free bit in K_3. The analysis is summarized in Table 8. As a result, 7 bits of the plaintext $(L_{21} \| R_{21})$ are influenced, and thus we need at most 2^7 chosen plaintexts. We can optimize the data complexity more. It is easy to imagine that giving influence to 6 bits of L_{21} by a carry from the change of 1 bit in R_{21} is hard. Moreover, if we can set the internal state value appropriately to avoid the carry expansion, the number of influenced bits can be limited. We confirmed that the impacts to L_{21} can only be limited to 3 cases regarding the carry in the addition of R_{22} and $F(L_{22}) \oplus (K_{\pi(22)} + \delta_{22})$; 1) the carry will not occur, 2) the carry (impact of the positive sign) will occur, and 3) the borrow (impact of the negative sign) will occur. Hence, by combining 1 bit impact in R_{21}, $3 \times 2 = 6$ chosen plaintexts is enough to perform the attack.

The attack is almost the same as the 29-round known-key attack except for the use of the splice-and-cut technique. The complexity slightly changes because the independent computation part becomes longer by 2 steps, which results in $2^{127.54}$ XTEA computations. The memory cost is for loading 6 chosen plaintexts and ciphertexts, which is 2^3 XTEA state.

7 Concluding Remarks

We presented the 3-subset meet-in-the-middle attack on XTEA. We firstly corrected the procedure for using the splice-and-cut technique to keep the data complexity small. We then attacked reduced-round XTEA by using this technique. With a non-marginal improvement, 25 rounds are attacked with 9 known plaintexts and 28 rounds are attacked 2^{37} chosen plaintexts. If a marginal improvement is allowed, the attack reached 29 rounds with 2 known plaintexts, and 31 rounds with 6 chosen plaintexts. As far as we know, our attacks are best in terms of the number of attacked rounds as a single-key attack.

References

1. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki, and Lei Wang. Preimages for step-reduced SHA-2. *Advances in Cryptology — ASIACRYPT 2009*, volume 5912 of LNCS, pages 578–597, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
2. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. *Advances in Cryptology — CRYPTO 2009*, volume 5677 of LNCS, pages 70–89, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
3. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. *Selected Areas in Cryptography SAC 2008*, volume 5381 of LNCS, pages 103–119, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
4. Eli Biham, Orr Dunkelman, Nathan Keller, and Adi Shamir. New data-efficient attacks on reduced-round IDEA. Cryptology ePrint Archive, Report 2011/417, 2011. http://eprint.iacr.org/2011/417.
5. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. *Advances in Cryptology — ASIACRYPT 2011*, volume 7073 of LNCS, pages 344–371, Berlin, Heidelberg, New York, 2011. Springer-Verlag.
6. Andrey Bogdanov and Christian Rechberger. A 3-subset meet-in-the-middle attack: Cryptanalysis of the lightweight block cipher KTANTAN. *Selected Areas in Cryptography SAC 2010*, volume 6544 of LNCS, pages 229–240, Berlin, Heidelberg, New York, 2011. Springer-Verlag.
7. Charles Bouillaguet, Orr Dunkelman, Gaëtan Leurent, and Pierre-Alain Fouque. Another look at complementation properties. *Fast Software Encryption 2010*, volume 6147 of LNCS, pages 347–364, Berlin, Heidelberg, New York, 2010.
8. Jiazhe Chen, Meiqin Wang, and Bart Preneel. Impossible differential cryptanalysis of the lightweight block ciphers TEA, XTEA and HIGHT. Cryptology ePrint Archive, Report 2011/616, 2011. http://eprint.iacr.org/2011/616.
9. Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8-round AES. *Fast Software Encryption 2008*, volume 5086 of LNCS, pages 116–126, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
10. Whitfield Diffie and Martin E. Hellman. Exhaustive cryptanalysis of the NBS Data Encryption Standard. *Computer*, Issue: 6(10), 1977.
11. Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Sangjin Lee. Differential cryptanalysis of TEA and XTEA. *Information Security and Cryptology ICISC 2003*, volume 2971 of LNCS, pages 402–417, Berlin, Heidelberg, New York, 2004. Springer-Verlag.
12. Takanori Isobe. A single-key attack on the full GOST block cipher. *Fast Software Encryption 2011*, volume 6733 of LNCS, pages 290–305, Berlin, Heidelberg, New York, 2011. Springer-Verlag.
13. Jens-Peter Kaps. Chai-Tea, cryptographic hardware implementations of xTEA. *INDOCRYPT 2008*, volume 5365 of LNCS, pages 363–375, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
14. John Kelsey, Bruce Schneier, and David Wagner. Key-schedule cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. *Advances in Cryptology — CRYPTO 1996*, volume 1109 of LNCS, pages 237–251, Berlin, Heidelberg, New York, 1996. Springer-Verlag.
15. John Kelsey, Bruce Schneier, and David Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. *ICICS 1997*, vol-

ume 1334 of LNCS, pages 233–246, Berlin, Heidelberg, New York, 1997. Springer-Verlag.

16. Youngdai Ko, Seokhie Hong, Wonil Lee, Sangjin Lee, and Ju-Sung Kang. Related key differential attacks on 27 rounds of XTEA and full-round GOST. *Fast Software Encryption 2004*, volume 3017 of LNCS, pages 299–316, Berlin, Heidelberg, New York, 2004.
17. Eunjin Lee, Deukjo Hong, Donghoon Chang, Seokhie Hong, and Jongin Lim. A weak key class of XTEA for a related-key rectangle attack. *VIETCRYPT*, volume 4341 of LNCS, pages 286–297, Berlin, Heidelberg, New York, 2006. Springer-Verlag.
18. Jiqiang Lu. Related-key rectangle attack on 36 rounds of the XTEA block cipher. *Int. J. Inf. Sec.*, 8(1):1 – 11, 2009.
19. Dukjae Moon, Kyungdeok Hwang, Wonil Lee, Sangjin Lee, and Jongin Lim. Impossible differential cryptanalysis of reduced round XTEA and TEA. *Fast Software Encryption 2002*, volume 2365 of LNCS, pages 49–60, Berlin, Heidelberg, New York, 2002. Springer-Verlag.
20. Roger M. Needham and David J. Wheeler. TEA extensions. Technical report, Computer Laboratory, University of Cambridge, October 1997.
21. Roger M. Needham and David J. Wheeler. Correction to xtea. Technical report, Computer Laboratory, University of Cambridge, October 1998. http://www.movable-type.co.uk/scripts/xxtea.pdf.
22. Markku-Juhani O. Saarinen. Cryptanalysis of Block-TEA. unpublished manuscript, 1998. http://groups.google.com/group/sci.crypt.research/msg/f52a533d1e2fa15e.
23. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of LNCS, pages 253–271, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
24. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of LNCS, pages 134–152, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
25. Gautham Sekar, Nicky Mouha, Vesselin Velichkov, and Bart Preneel. Meet-in-the-middle attacks on reduced-round XTEA. *CT-RSA 2011*, volume 6558 of LNCS, pages 250–267, Berlin, Heidelberg, New York, 2011. Springer-Verlag.
26. Michael Steil. 17 mistakes microsoft made in the Xbox security system. 22nd Chaos Communication Congress, 2005. http://events.ccc.de/congress/2005/fahrplan/events/559.en.html.
27. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved meet-in-the-middle cryptanalysis of KTANTAN. Cryptology ePrint Archive, Report 2011/201, 2011. http://eprint.iacr.org/2011/201.
28. Lei Wei, Christian Rechberger, Jian Guo, Hongjun Wu, Huaxiong Wang, and San Ling. Improved meet-in-the-middle cryptanalysis of KTANTAN (Poster). *ACISP 2011*, volume 6812 of LNCS, pages 433–438, Berlin, Heidelberg, New York, 2011. Springer-Verlag.
29. David J. Wheeler and Roger M. Needham. TEA, a Tiny Encryption Algorithm. *Fast Software Encryption 1994*, volume 1008 of LNCS, pages 363–366, Berlin, Heidelberg, New York, 1995.
30. Elias Yarrkov. Cryptanalysis of XXTEA. Cryptology ePrint Archive, Report 2010/254, 2010. http://eprint.iacr.org/2010/254.