Improving the Biclique Cryptanalysis of AES

Biaoshuai Tao(✉) and Hongjun Wu(✉)

Nanyang Technological University, Singapore, Republic of Singapore
taob0001@e.ntu.edu.sg, wuhj@ntu.edu.sg

Abstract. Biclique attack is currently the only key-recovery attack on the full AES with a single key. Bogdanov *et al.* applied it to all the three versions of AES by constructing bicliques with size $2^8 \times 2^8$ and reducing the number of S-boxes computed in the matching phase. Their results were improved later by better selections of differential characteristics in the biclique construction. In this paper, we improve the biclique attack by increasing the biclique size to $2^{16} \times 2^8$ and $2^{16} \times 2^{16}$. We have a biclique attack on each of the following AES versions:

– AES-128 with time complexity $2^{126.13}$ and data complexity 2^{56},
– AES-128 with time complexity $2^{126.01}$ and data complexity 2^{72},
– AES-192 with time complexity $2^{189.91}$ and data complexity 2^{48}, and
– AES-256 with time complexity $2^{254.27}$ and data complexity 2^{40}.

Our results have the best time complexities among all the existing key-recovery attacks with data less than the entire code book.

Keywords: AES · Biclique attack · Large biclique

1 Introduction

Rijndael was selected as the Advanced Encryption Standard (AES) in 2001 [13]. AES is widely used today since it is strong in security and efficient in both software and hardware. AES was designed to resist differential, linear cryptanalysis and square attacks. The best impossible differential attack (first introduced in [2]) can break 7-round AES-128 [22], and the best square attack can break 8-round AES-192 [14]. The recent meet-in-the-middle (MITM) attack can break 9-round AES-192 and AES-256 [20]. The related-key attacks are powerful against AES [3–6,15,17], but the related-key attacks have limited impact on the security of AES when the secret keys in AES are generated securely.

The first key-recovery attack on the full AES with single key is the *biclique attack* [9]. Biclique attack on AES is faster than brute force by a factor of three to four. Biclique attack can be considered as an enhanced version of the meet-in-the-middle (MITM) attack. It was first introduced in the preimage attacks against hash functions Skein and SHA-2 [19]. After being applied to AES, the biclique attack was applied to attack other block ciphers, such as ARIA [1,11], Hight [16], IDEA [18], Piccolo [23], SQUARE [21], Twine [12].

Although the idea in [9] is revolutionary, both the time complexities and the data complexities of the attacks in [9] have room for improvement by better selections of differential characteristics in the biclique construction. This has been illustrated in [1,7,8].

The independent-biclique paradigm can be further improved in time complexity by the *sieve-in-the-middle* (SIM) technique [10], a clever way to further speed up the matching computation by precomputing and storing the possible transitions for the middle superbox in a table of size 2^{32}, but with the extra cost of accessing large lookup tables is introduced.

In this paper, we improve the independent-biclique paradigm in [9] by using larger bicliques. We improve the time complexities for all the three AES versions. We also obtain improvements in data complexity for AES-128. Our results are summarized in Table 1 with comparison to the previous results. Among all the existing attacks which have data complexities that are less than 2^{128}, our results have minimum time complexities. Notice that our results can be naturally combined with sieve-in-the-middle (SIM) technique as shown in Table 1, which is further discussed in Sect. 7.

Table 1. Summary of our results

algorithm	data	computation without SIM	memory in bytes	computation with SIM	memory in bytes	reference
AES-128 previous results	2^{88}	$2^{126.21}$	$2^{14.32}$	-	-	[9]
	2^{4}	$2^{126.89}$	$2^{14.32}$	-	-	[8]
	2^{72}	$2^{126.72}$	$2^{14.32}$	-	-	[1]
	-	-	-	$2^{126.01}$ $(2^{125.69})$[1]	2^{64}	[10]
	2	$2^{126.67}$	$2^{14.32}$	$2^{126.59}$	2^{64}	[7][2]
	2^{64}	$2^{126.16}$	$2^{14.32}$	$2^{126.01}$	2^{64}	[7]
AES-128 our results	2^{56}	$2^{126.13}$	$2^{22.07}$	$2^{125.99}$	2^{64}	Sect. 4
	2^{72}	$2^{126.01}$	$2^{26.14}$	$2^{125.87}$	2^{64}	Appendix A
AES-192 previous results	2^{80}	$2^{190.16}(2^{189.74})$[3]	$2^{14.39}$	-	-	[9]
	2^{48}	$2^{190.28}$	$2^{14.39}$	-	-	[1]
	-	-	-	$2^{190.04}$	2^{64}	[10]
	2	$2^{190.9}$	$2^{14.39}$	$2^{190.83}$	2^{64}	[7]
	2^{48}	$2^{190.16}$	$2^{14.39}$	$2^{190.05}$	2^{64}	[7]
AES-192 our results	2^{48}	$2^{189.91}$	$2^{22.27}$	$2^{189.76}$	2^{64}	Appendix B
AES-256 previous results	2^{40}	$2^{254.58}(2^{254.42})$[4]	$2^{14.54}$	-	-	[9]
	2^{64}	$2^{254.53}$	$2^{14.54}$	-	-	[1]
	-	-	-	$2^{254.51}$	2^{64}	[10]
	3	2^{255}	$2^{14.54}$	$2^{254.94}$	2^{64}	[7]
	2^{40}	$2^{254.31}$	$2^{14.54}$	$2^{254.24}$	2^{64}	[7]
AES-256 our results	2^{40}	$2^{254.27}$	$2^{22.61}$	$2^{254.18}$	2^{64}	Appendix B

[1] Our analysis shows that the time complexity to be $2^{126.01}$ instead of $2^{125.69}$ claimed in [10], based on the same criteria used in Sect. 4.5.

[2] The results with data complexity 2^{128} in [7] are not shown here, as we do not discuss the attack with entire code book in this paper.

[3] The accurate result is $2^{190.16}$ instead of $2^{189.74}$ originally claimed, as already corrected in [1].

[4] The accurate result is $2^{254.58}$ instead of $2^{254.42}$ originally claimed, as already corrected in [7].

This paper is organized as follows. The biclique attack on AES is introduced in Sect. 2. Section 3 gives an overview of our biclique attack on AES. Our detailed biclique attack on AES-128 is given in Sect. 4. Section 5 gives comparisons with our results to the previous ones. We verify our results in Sect. 6, and combine our results to the sieve-in-the-middle technique in Sect. 7. Section 8 concludes the paper.

2 The Biclique Attack

In this section, we give the general description of the biclique attack against block ciphers.

2.1 The Biclique

Consider a block cipher e which maps plaintext P to ciphertext C. Assume that the cipher can be decomposed as $e = g_2 \circ f \circ g_1$:

$$e : P \xrightarrow[g_1]{} S \xrightarrow[f]{} T \xrightarrow[g_2]{} C.$$

Consider 2^{d_1} intermediate states $\{S_i : i = 0, 1, \ldots, 2^{d_1} - 1\}$, 2^{d_2} intermediate states $\{T_j : j = 0, 1, \ldots, 2^{d_2} - 1\}$ and $2^{d_1+d_2}$ keys $\{K[i,j]\}$. The 3-tuple $(\{S_i\}, \{T_j\}, \{K[i,j]\})$ is called a *biclique*, if $S_i \xrightarrow[f]{K[i,j]} T_j$ for all $i \in \{0, 1, \ldots, 2^{d_1} - 1\}$ and all $j \in \{0, 1, \ldots, 2^{d_2} - 1\}$. Here $K[i,j]$ maps the state S_i to the state T_j by the subcipher f.

In most of the attacks against AES as well as ours, bicliques are constructed "at the end", in which case g_2 is the identity map and $\{T_j\}$ becomes ciphertexts set $\{C_j\}$. Correspondingly, the biclique takes the form $(\{S_i\}, \{C_j\}, \{K[i,j]\})$. Bicliques "in the middle" are also considered in [7]. However, attacks based on such bicliques usually require extremely high data complexities, as the differences in $\{T_j\}$ will propagate to ciphertexts (refer to [7] for examples in AES).

Define the *length* of a biclique be the number of rounds covered by f, and the *size* be $2^{d_1} \times 2^{d_2}$. In the original attack [9] and most of the subsequent improvements, bicliques of size $2^8 \times 2^8$ are constructed in all the three versions of AES. In [7], bicliques of size $2^{16} \times 1$, i.e. stars, are also considered. The attack with stars only requires minimal data, but the time complexity becomes higher (refer to Table 1, the rows with data complexities 2 or 3). In our attack, the size of bicliques is enlarged to $2^{16} \times 2^8$, and even to $2^{16} \times 2^{16}$ for AES-128.

2.2 Outline of the Biclique Attack

As mentioned in the last section, we focus on the attack with g_2 being the identity map in which the biclique is "at the end". In this case, the biclique attack can be sketched as follows:

1. **Partition**: Partition 2^n keys into $2^{n-(d_1+d_2)}$ sets of size $2^{d_1+d_2}$.

2. **Biclique Construction**: For each partition $\{K[i,j]\}$, construct biclique $(\{S_i\},\{C_j\},\{K[i,j]\})$.
3. **Oracle Decryption**: Decrypt 2^{d_2} ciphertexts $\{C_j\}$ by the oracle with the secret key to obtain the corresponding 2^{d_2} plaintexts $\{P_j\}$.
4. **Matching**: for each pair (P_j, S_i), if $P_j \xrightarrow[g_1]{K[i,j]} S_i$, $K[i,j]$ is a candidate.

In the following sections, we introduce in details how the bicliques can be constructed, and how we can reduce the time complexity for the matching step (Step 4) by using a technique called *matching with precomputation* in [9].

2.3 Constructing Bicliques from Independent Related-Key Differentials

Two biclique attack paradigms are shown in [9], long biclique and independent biclique. We will only focus on independent biclique, as long-biclique-based attack currently cannot work for full AES.

Fix a tuple $(S_0, C_0, K[0,0])$ where $K[0,0]$ maps S_0 to C_0, and we aim to fill in the other $2^{d_1}-1$ intermediate states, $2^{d_2}-1$ ciphertexts and $2^{d_1+d_2}-1$ keys to get a biclique. Consider the following two sets of related-key differentials with respect to the base computation $S_0 \xrightarrow[f]{K[0,0]} C_0$:

– Δ_j**-differential:** It maps a zero input difference to an output difference Δ_j under a key difference Δ_j^K:
  $0 \xrightarrow[f]{\Delta_j^K} \Delta_j$ with $\Delta_0^K = 0$ and $\Delta_0 = 0$, where $j = 0, 1, \ldots, 2^{d_2}-1$;
– ∇_i**-differential:** It maps an input difference ∇_i to a zero output difference under a key difference ∇_i^K:
  $\nabla_i \xrightarrow[f]{\nabla_i^K} 0$ with $\nabla_0^K = 0$ and $\nabla_0 = 0$, where $i = 0, 1, \ldots, 2^{d_1}-1$.

According to [9], if the characteristics of Δ_j-differentials do not share active nonlinear components (which are S-boxes in our case of AES) with the characteristics of ∇_i-differentials, then the tuple $(S_0, C_0, K[0,0])$ will conform to all the $2^{d_1+d_2}$ combined (Δ_j, ∇_i)-differentials:

$$\nabla_i \xrightarrow[f]{\Delta_j^K \oplus \nabla_i^K} \Delta_j \text{ for } i \in \{0,1,\ldots,2^{d_1}-1\} \text{ and } j \in \{0,1,\ldots,2^{d_2}-1\},$$

which means

$$S_0 \oplus \nabla_i \xrightarrow[f]{K[0,0] \oplus \Delta_j^K \oplus \nabla_i^K} C_0 \oplus \Delta_j.$$

Finally, we obtain the biclique by putting

$$S_i = S_0 \oplus \nabla_i, \qquad C_j = C_0 \oplus \Delta_j \qquad \text{and} \qquad K[i,j] = K[0,0] \oplus \Delta_j^K \oplus \nabla_i^K,$$

where we require $\Delta_j^K \neq \nabla_i^K$ whenever $i + j > 0$, as we want all the $2^{d_1+d_2}$ keys in $\{K[i,j]\}$ to be distinct.

In the attack, an attacker first finds the key group $\{K[i,j]\}$ satisfying the above independent differentials property, then computes all the C_j from S_0 by Δ_j-differentials, and all the S_i from C_0 by ∇_i-differentials. This requires at most $2^{d_1} + 2^{d_2}$ computations of f.

In the case of the independent-biclique attack against AES, the cost of constructing a biclique turns out to be low, compared to the matching part (Step 4) which requires almost $2^{d_1+d_2}$ computations of g_1. Naturally, we aim to construct bicliques as long as possible in order to reduce the number of rounds of g_1. In [9], the biclique of length 3 is constructed for AES-128, and of length 4 for AES-192 and AES-256. Unfortunately, we cannot enlarge these lengths due to the extremely fast diffusion of AES, otherwise this will yield a decent improvement in time complexity. Instead, we increase the biclique size as mentioned. This reduces the time complexity in the matching part, as will be shown later.

2.4 Matching with Precomputations

In the last step, i.e. the matching step, an attacker needs to check whether P_j is mapped to S_i by the key $K[i,j]$ through g_1. To speed up the attack, instead of matching on a full state, an attacker considers matching variable v, which can be a single byte in the case of AES:

$$P_j \xrightarrow{K[i,j]} v \xleftarrow{K[i,j]} S_i.$$

This involves $2^{d_1+d_2}$ computations for g_1. However, for fixed j and two different i_1, i_2, some parts of the states in the forward computation $P_j \xrightarrow{K[i_1,j]} v$ and $P_j \xrightarrow{K[i_2,j]} v$ are still the same, for which we only need to compute once. It is similar for $v \xleftarrow{K[i,j_1]} S_i$ and $v \xleftarrow{K[i,j_2]} S_i$ in the backward computation. The *matching with precomputation* technique, introduced in [9], makes use of this observation. It precomputes and stores those parts that are neutral to different i values in forward direction, and those parts that are neutral to different j values in backward direction. As a result, we only need to recompute those unneutral parts in the matching step, and look up from the stored precomputation for neutral parts.

3 Overview of Our Biclique Attacks on AES

We construct bicliques with sizes $2^{16} \times 2^8$ and $2^{16} \times 2^{16}$ to improve the biclique attack against AES. In this section, we give an overview of our technique.

3.1 The Bicliques in Our Attacks

In our attacks, the state sets $\{S_i\}$ are increased by a factor of 2^8:

$$(\{S_{i_1,i_2}\}, \{C_j\}, \{K[i_1,i_2,j]\}) \text{ for all } i_1, i_2, j \in \{0, 1, \ldots, 2^8-1\},$$

where

$$S_{i_1,i_2} \xrightarrow[f]{K[i_1,i_2,j]} C_j.$$

Note that this biclique is now of size $2^{16} \times 2^8$:

$$|\{S_{i_1,i_2}\}| = 2^{16}, \qquad |\{C_j\}| = 2^8, \qquad \text{and} \qquad |K[i_1,i_2,j]| = 2^{24}.$$

To construct such a biclique, we first fix the base computation $S_{0,0} \xrightarrow[f]{K[0,0,0]} C_0$. We then look for

- Δ_j**-differentials** which maps input difference 0 to an output difference Δ_j;
- ∇_{i_1}**-differentials** and ∇_{i_2}**-differentials** which map input differences ∇_{i_1} and ∇_{i_2} respectively to the output difference 0.

To make the biclique valid, we only need to make sure that there is no common active nonlinear components, the S-boxes in the case of AES, in either pair of differential characteristics: (Δ_j, ∇_{i_1}) and (Δ_j, ∇_{i_2}). To get the actual biclique, we need 3×2^8 computations of f for each of the three differentials above.

We have also designed a second attack specifically for AES-128 which considers even larger biclique $(\{S_{i_1,i_2}\}, \{C_{j_1,j_2}\}, \{K[i_1,i_2,j_1,j_2]\})$, in which the size of the ciphertext sets $\{C_j\}$ is further increased by a factor of 2^8. Correspondingly, we look for $\Delta_{j_1}, \Delta_{j_2}, \nabla_{i_1}, \nabla_{i_2}$ differentials such that none of $(\Delta_{j_1}, \nabla_{i_1})$, $(\Delta_{j_2}, \nabla_{i_1})$, $(\Delta_{j_1}, \nabla_{i_2})$ and $(\Delta_{j_2}, \nabla_{i_2})$ shares active nonlinear components. As a result, a total of 4×2^8 computations of f is needed.

3.2 Less Parts Being Recomputed in the Matching Step

After decrypting each C_j by the oracle, we obtain 2^8 plaintexts $\{P_j\}$. In the matching step, we apply subcipher g_1 to each P_j with key $K[i_1,i_2,j]$ to check whether we could get exactly S_{i_1,i_2}. If so, $K[i_1,i_2,j]$ is proposed as a candidate. We again match only the matching variable v:

$$P_j \xrightarrow{K[i_1,i_2,j]} v \xleftarrow{K[i_1,i_2,j]} S_{i_1,i_2}.$$

In the precomputation phase, we store the following 3×2^{16} computations:

Computation 1. $P_j \xrightarrow{K[0,i_2,j]} v$ for $i_2, j \in \{0,1,\ldots,2^8-1\}$,
Computation 2. $P_j \xrightarrow{K[i_1,0,j]} v$ for $i_1, j \in \{0,1,\ldots,2^8-1\}$,
Computation 3. $v \xleftarrow{K[i_1,i_2,0]} S_{i_1,i_2}$ for $i_1, i_2 \in \{0,1,\ldots,2^8-1\}$.

Same as in Sect. 2.4, in the recomputation phase when we match P_j to S_{i_1,i_2} on v, we only need to recompute those unneutral parts. The advantage of our technique is that less parts are needed to be recomputed. In the forward computation, we only need to recompute those parts which are unneutral to *both* i_1 and i_2. In other words, we can look up from two computations (Computation

1 and Computation 2 above) for the forward recomputation, instead of only one in the original attack, which causes less parts being recomputed and thus less time complexity in recomputation phase. Since the recomputation phase is the computational bottleneck, this advantage will reduce the total time complexity.

In our second attack on AES-128 with the biclique of size $2^{16} \times 2^{16}$ taking the form $(\{S_{i_1,i_2}\}, \{C_{j_1,j_2}\}, \{K[i_1,i_2,j_1,j_2]\})$, we decrypt each C_{j_1,j_2} to get P_{j_1,j_2}. In this case, we need to store 4×2^{24} computations in the precomputation phase:

Computation 1. $P_{j_1,j_2} \xrightarrow{K[0,i_2,j_1,j_2]} v$ for $i_2, j_1, j_2 \in \{0, 1, \ldots, 2^8 - 1\}$,
Computation 2. $P_{j_1,j_2} \xrightarrow{K[i_1,0,j_1,j_2]} v$ for $i_1, j_1, j_2 \in \{0, 1, \ldots, 2^8 - 1\}$,
Computation 3. $v \xleftarrow{K[i_1,i_2,0,j_2]} S_{i_1,i_2}$ for $i_1, i_2, j_2 \in \{0, 1, \ldots, 2^8 - 1\}$,
Computation 4. $v \xleftarrow{K[i_1,i_2,j_1,0]} S_{i_1,i_2}$ for $i_1, i_2, j_1 \in \{0, 1, \ldots, 2^8 - 1\}$.

The time complexity is further reduced as we also have two different references for the backward recomputation to looked up, namely, Computation 3 and 4. However, this improvement in time complexity does have disadvantages. Besides the obviously higher cost of memory, this attack may also pay extra cost on data complexity, as the additional Δ_{j_2}-differential may corrupt the extra neutral bytes in ciphertext.

4 The Improved Biclique Attacks Against AES

In this section, we describe our biclique attack against AES-128 by using the $2^{16} \times 2^8$ biclique. The attack on AES-128 with $2^{16} \times 2^{16}$ biclique is given in Appendix A, while the attacks on AES-192 and AES-256 with $2^{16} \times 2^8$ bicliques are illustrated with figures in Appendix B. Refer to Table 1 for all the results.

In this section and the two Appendix sections, we will use $\$0, \$1, \$2, \ldots$ to denote the round keys. Bytes within a state and a round key are enumerated as follow, and byte i in state Q is denoted as Q_i.

0	4	8	12
1	5	9	13
2	6	10	14
3	7	11	15

4.1 Key Partitioning

We define the key space with respect to the round key \$8 (the same as that in [9]). This definition is valid as the AES-128 key schedule bijectively maps each key to \$8. The base keys $K[0,0,0]$ are all the possible 2^{104} 16-byte values with three bytes, $\$8_0, \$8_1, \$8_6$, fixed to 0, whereas the remaining 13 bytes run over all values. The keys $\{K[i_1,i_2,j]\}$ in a group are enumerated by all possible byte differences i_1, i_2 and j with respect to the base key $K[0,0,0]$. The space of the round key \$8 (and hence the AES key space) is thus partitioned into 2^{104} sets of size 2^{24}.

0			
0			
	0		

i_1		i_1	
i_2		i_2	
	j		

4.2 3-round Biclique of Size $2^{16} \times 2^8$

In the next step, we construct a 3-round biclique by following the steps in Sect. 3. Figure 1 illustrates the $\Delta_j, \nabla_{i_1}, \nabla_{i_2}$ differentials, from which we can easily verify that (Δ_j, ∇_{i_1}) and (Δ_j, ∇_{i_2}) shares no common active S-box.

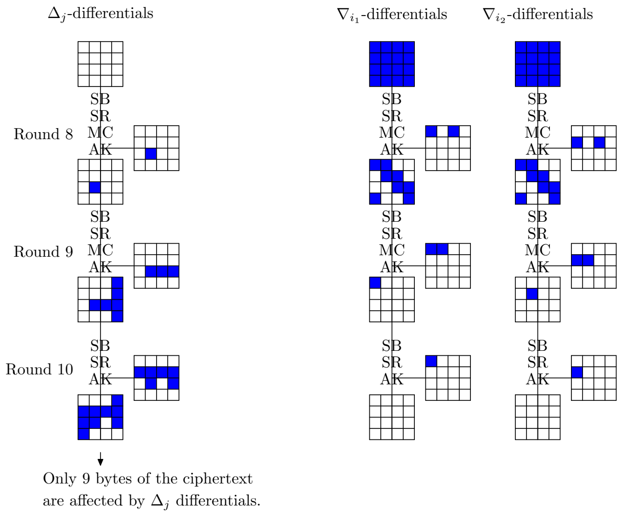

Δ_j-differentials
∇_{i_1}-differentials
∇_{i_2}-differentials
Round 8
Round 9
Round 10
SB
SR
MC
AK
Only 9 bytes of the ciphertext are affected by Δ_j differentials.

Fig. 1. All the $\Delta_j, \nabla_{i_1}, \nabla_{i_2}$ differentials: AES-128

4.3 Computing Round Keys

Now for each round key $K[i_1, i_2, j]$ of round 8, we calculate its corresponding round keys $\$0, \$1, \ldots, \$7$. At the first glance, it requires 2^{24} operations of the AES key schedule. However, the calculation can be speed up with the precomputation technique introduced earlier. We will apply this technique only on the first column of each round key, as the rest three columns require only xor operations to be computed which has negligible time complexity.

In the precomputation phase, we compute all the 8 round keys for the following:

- $K[0, i_2, j]$ for all $i_2, j \in \{0, 1, \ldots, 2^8 - 1\}$;
- $K[i_1, 0, j]$ for all $i_1, j \in \{0, 1, \ldots, 2^8 - 1\}$.

This requires at most 2×2^{16} 8-round computations of key schedule.

In the recomputation phase when we are calculating $K[i_1, i_2, j]$ (for all the 8 round keys from \$7 all the way down to \$0), we only need cheap xor and lookup operations (no S-box operation in particular) by storing all the bytes with $*$ and the S-box substitution values of all the bytes with $\circ$ in Fig. 2.

For example, to compute the first byte of \$7, we have $\$7_0 = \$8_0 \oplus s(\$7_{13}) \oplus roundConst$, where $s(\$7_{13})$ has the same value for $K[0, i_2, j]$ and $K[i_1, i_2, j]$ and we already store the one for $K[0, i_2, j]$. For the second and the third bytes $\$7_1, \$7_2$, we look them up directly from $K[0, i_2, j]$, and we lookup from $K[i_1, 0, j]$ for the forth byte $\$7_3$. After we obtain the first column of \$7, we compute the rest 12 bytes and get the entire \$7. After we recover the full \$7, we do the same to compute \$6 and so on.

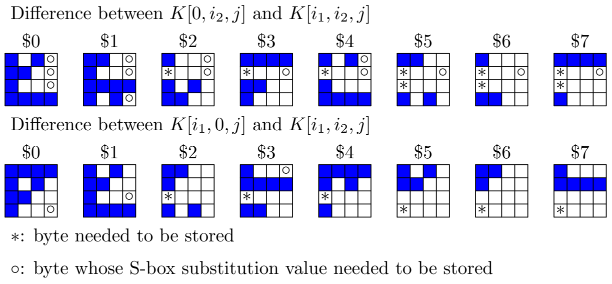

Difference between $K[0, i_2, j]$ and $K[i_1, i_2, j]$
\$0
\$1
\$2
\$3
\$4
\$5
\$6
\$7
Difference between $K[i_1, 0, j]$ and $K[i_1, i_2, j]$
\$0
\$1
\$2
\$3
\$4
\$5
\$6
\$7
$*$: byte needed to be stored
$\circ$: byte whose S-box substitution value needed to be stored

Fig. 2. Precomputation and recomputation of round keys: AES-128

4.4 Matching Over 7 Rounds

Now we check whether the secret key belongs to the key group $\{K[i_1, i_2, j]\}$. As shown in Fig. 3 and Fig. 4, we match the first byte in the state after Round 2, which is the same as [9]. So we only need to compute 4 bytes in Round 2, 1 byte in Round 3 and 4 bytes in Round 4. Additionally, benefiting from the matching with precomputation technique (Sect. 2.4), we only need to compute very few bytes in Round 1 and Round 7 in the recomputation phase. Specifically, we only need to compute 5 bytes in Round 1 (Fig. 3) and 8 bytes in Round 7 (Fig. 4).

4.5 Complexity of the Attack

Now we evaluate the time complexity, data complexity and memory complexity of our attack.

Time Complexity. Similar to the original biclique attack [9], we count the number of S-boxes computations to determine the time complexity. Note that there are 200 S-boxes in one full AES-128 (160 for encryption/decryption, 40 for key schedule). We measure the time complexity in terms of the full AES-128 operations, so the overall time complexity will be given as the total number of S-boxes in our attack divided by 200.

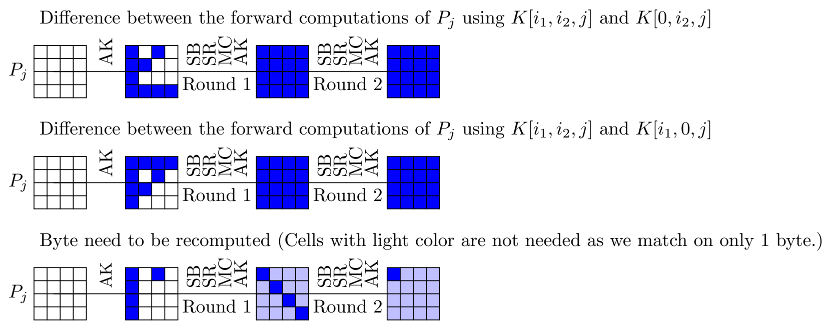

Difference between the forward computations of P_j using $K[i_1, i_2, j]$ and $K[0, i_2, j]$
P_j
AK
SB SR MC AK
Round 1
SB SR MC AK
Round 2
Difference between the forward computations of P_j using $K[i_1, i_2, j]$ and $K[i_1, 0, j]$
P_j
AK
SB SR MC AK
Round 1
SB SR MC AK
Round 2
Byte need to be recomputed (Cells with light color are not needed as we match on only 1 byte.)
P_j
AK
SB SR MC AK
Round 1
SB SR MC AK
Round 2

Fig. 3. Forward recomputation in matching: AES-128

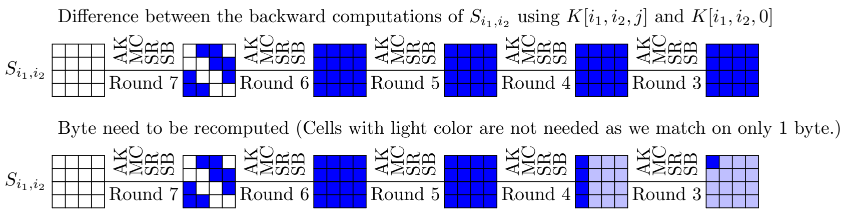


Fig. 4. Backward recomputation in matching: AES-128

For the search in a single key group $\{K[i_1, i_2, j]\}$, the time complexity consists of the following components:

1. C_{biclique}: The complexity of constructing the biclique. (Sect. 4.2)
2. C_{oracle}: The complexity for the decryption of each C_j by the oracle.
3. C_{keys}: The complexity of computing all the round keys for key $K[i_1, i_2, j]$ in the set $\{K[i_1, i_2, j]\}$. (Sect. 4.3)
4. C_{prec}: The complexity of precomputation phase. (Sect. 4.4)
5. C_{rec}: The complexity of recomputation phase. (Sect. 4.4)
6. C_{falsep}: The complexity generated from false positives.

The time complexity for biclique construction is merely the 3×2^8 computations of 3 rounds subcipher f which is 56 S-boxes (including 2 rounds key schedule). This complexity is given as

$$C_{\text{biclique}} = 3 \times 2^8 \times 56/200 = 2^{7.75}.$$

We need to decrypt all the 2^8 ciphertexts $\{C_j\}$:

$$C_{\text{oracle}} = 2^8.$$

The time complexity of computing all the round keys includes the precomputation phase and recomputation phase. As mentioned in Sect. 4.3, the recomputation of round keys require only xor and lookup operations, which correspond to 0 S-box operation. As for precomputation, we need to compute those 8 round keys (equivalent to 32 S-boxes) for both groups $K[0, i_2, j]$ and $K[i_1, 0, j]$, and each group is of size 2^{16}. The time complexity of round key computation is

$$C_{\text{keys}} = 2 \times 2^{16} \times 32/200 = 2^{14.36}.$$

In the precomputation phase of matching, we only need to compute one round for Computation 1, 2 and 3 in Sect. 3.2. This is because all the 16 bytes of each state become different except for Round 1 and 7 (refer to Fig. 3 and Fig. 4), so the precomputation of other rounds provides no information for the recomputation at all, which we do not need to store. Since exactly 20 S-boxes are included in a round, we have

$$C_{\text{prec}} = 3 \times 2^{16} \times 20/200 = 2^{14.26}.$$

In the recomputation state, as shown in Fig. 3 and 4, we need to recompute 9 S-boxes in the forward direction and 45 S-boxes in the backward direction. There are 54 S-boxes in total, and it is needed for each of the 2^{24} keys $\{K[i_1, i_2, j]\}$.

$$C_{\text{rec}} = 2^{24} \times 54/200 = 2^{22.11}.$$

We performed 2^{24} matchings on a single byte with 2^8 possible values, the number of false positives is approximately $2^{24-8} = 2^{16}$. We eliminate the false positives by matching them on a full 16-byte state, which require 3 rounds computation (i.e. Round 2, 3 and 4), so 48 S-boxes are needed.

$$C_{\text{falsep}} = 2^{16} \times 48/200 = 2^{13.94}.$$

Summing up all the above, we have the total time complexity:

$$2^{104}(2^{7.75} + 2^8 + 2^{14.36} + 2^{14.26} + 2^{22.11} + 2^{13.94}) = 2^{126.13}.$$

Note that with the above time complexity, the secret key of AES-128 is obtained with success rate 1.

Data Complexity. According to Fig. 1, Δ_j-differential affect only 9 bytes of the ciphertext, and all the ciphertexts have constant values at bytes $C_{0,4,7,8,10,11,15}$. Furthermore, the ciphertext bytes C_1, C_5 and C_{13} of those 9 bytes always maintain the same difference. As a result, the data complexity does not exceed $2^{(9-3+1)\times 8} = 2^{56}$.

Memory Complexity. We need to store the biclique, as well as the precomputations for both round keys and states. Note that we do not need to store all the 2^{24} sets of round keys, as it can be computed in runtime whenever needed.

For the biclique storage, we need to store 2^{16} states $\{S_{i_1,i_2}\}$ and 2^8 ciphertexts $\{C_j\}$, with size 16 bytes for each. The total memory complexity for biclique storage is $(2^8 + 2^{16}) \times 16$ bytes.

For the precomputations for round keys and states, we only store the bytes that are looked up in the recomputation stage (those neutral bytes not affected by the differential charactertistic). For the round key precomputation, as shown in Fig. 2, we need to store a total of $2^{16} \times 32$ bytes (all the $*$ and $\circ$) which includes $2^{16} \times 23$ bytes from $K[0, i_2, j]$ and $2^{16} \times 9$ bytes from $K[i_1, 0, j]$.

Similarly, for the storage of states used in the matching, we only need to store 11 bytes in the state after the SubBytes operation in Round 1, and 8 bytes in the state between Round 6 and Round 7, which is $2^{16}(11+8) = 2^{16} \times 19$ (bytes).

Finally, the total memory complexity is

$$(2^8 + 2^{16}) \times 16 + 2^{16} \times 32 + 2^{16} \times 19 = 2^{22.07} \text{ (bytes)}.$$

We used the same criteria to analyze the memory complexities for all the previous attacks as well, and obtain these data in Table 1 (the forth column).

5 Comparing with the Previous Biclique Attacks

As mentioned in Sect. 3.2, the main advantage of our technique is reducing the number of S-boxes being recomputed in the matching step. Table 2 compares the number of S-boxes needed in the recomputation, and it shows that our attack requires least S-boxes to be recomputed (with data complexity less than the full code book).

Table 2. Comparison of the numbers of S-boxes recomputed

	our results with $2^{16} \times 2^8$ bicliques	our results with $2^{16} \times 2^{16}$ bicliques	[9] (original attack)	[7] (prev. best results)
AES-128	54	50	57	55
AES-192	52	-	61	62
AES-256	83	-	101	86

It is illuminating to compare the two results for AES-128. With larger bicliques, the number of S-boxes computed can be further reduced from 54 to 50 which leads an improvement of time complexity from $2^{126.13}$ to $2^{126.01}$ (Table 1). However, by applying larger biclique, the data complexity and memory complexity increase significantly (Table 1).

6 Verification in Experiment

In our attack against all the three AES versions, the whole key space is partitioned into $2^{104}/2^{168}/2^{232}$ sets with sizes 2^{24} (or 2^{96} sets with size 2^{32} in the second attack of AES-128). We wrote a program to search the key in a single partition. The secret key is found if it was set in the searched partition. This program proves the correctness of our attack algorithm, including the validity of bicliques and the correctness of the differential characteristics. The program also counts the number of S-boxes calculated in the search of a full partition, and outputs the time complexities in terms of equivalent full AES computations. The memory complexity is also verified in the experiment by checking the memory usage of the process. We verified the attacks on all the three versions of AES, and the experimental results are even slightly better than our theoretical results.

7 Combination with the Sieve-in-the-Middle Technique

Sieve-in-the-Middle (SIM) technique by Canteaut et al. [10] is a variant of the Meet-in-the-Middle (MITM) technique. In its application to the biclique attack against AES, it can save another 5 S-boxes in recomputation during the matching step, by storing 2^{32} lookup tables of size 2^{32} bytes each. With a large increment in memory complexities, all the numbers shown in Table 2 can be reduced by 5, which results in further improvements in time complexities (see Table 1). The reader can also refer to Sect. 8 of [7] for more details.

8 Conclusion

In this paper, we improved the independent-biclique paradigm of the biclique attack [9] by increasing the biclique size. Our technique enhances the matching-with-precomputation technique in [9] by reducing the number of S-boxes being recomputed. The data complexities in the attacks against AES-128 are also reduced. Our attacks are currently the fastest with moderate data complexities.

A Biclique Attack on AES-128 with $2^{16} \times 2^{16}$ Biclique

A.1 Key Partitioning

We again define the key space with respect to the round key \$8. Fix $\$8_0, \$8_1, \$8_7$ and $\$8_8$ to 0. The keys $\{K[i_1, i_2, j_1, j_2]\}$ in a group are enumerated by all possible byte differences i_1, i_2, j_1 and j_2 with respect to the base key $K[0, 0, 0, 0]$. The AES key space is thus partitioned into the 2^{96} sets of size 2^{32}.

0		0	
0			
	0		

i_1		$i_1 \oplus j_1$	
i_2		i_2	
	j_2		

The fact that $\$8_8$ is shared by the byte differences i_1 and j_1 does not invalidate the biclique, as this shared difference has not passed into any non-linear S-box operation (refer to Fig. 5).

A.2 3-round Biclique of Size $2^{16} \times 2^{16}$

Figure 5 illustrates the $\Delta_{j_1}, \Delta_{j_2}, \nabla_{i_1}, \nabla_{i_2}$ differentials, from which we can easily verify that each of the two Δ differentials share no active S-boxes to each of the two ∇ differentials.

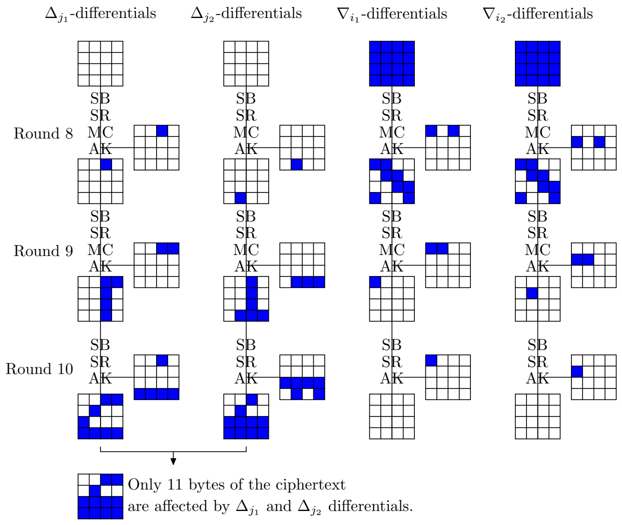

Δ_{j_1}-differentials
Δ_{j_2}-differentials
∇_{i_1}-differentials
∇_{i_2}-differentials
Round 8
Round 9
Round 10
SB
SR
MC
AK
Only 11 bytes of the ciphertext are affected by Δ_{j_1} and Δ_{j_2} differentials.

Fig. 5. All the $\Delta_{j_1}, \Delta_{j_2}, \nabla_{i_1}, \nabla_{i_2}$ differentials: AES-128

A.3 Computation of Round Keys

For each round key $K[i_1, i_2, j_1, j_2]$ of round 8, we calculate its corresponding round keys \$0, \$1, . . . , \$7. In the precomputation phase, we compute and store all the 8 round keys for the following:

– $K[0, i_2, j_1, j_2]$ for all $i_2, j_1, j_2 \in \{0, 1, \ldots, 2^8 - 1\}$;
– $K[i_1, 0, j_1, j_2]$ for all $i_1, j_1, j_2 \in \{0, 1, \ldots, 2^8 - 1\}$.

This requires at most $2 \cdot 2^{24}$ 8-round computations of key schedule.

Since we are using exactly the same $\nabla_{i_1}, \nabla_{i_2}$ differential characteristics, the key differential patterns of the above two computations are the same to those in Fig. 2. For the same reason, we only need to store those marked bytes, and the recomputation of round keys require no S-box computation. The only difference is that each $*$ or $\circ$ now corresponds to 2^{24} bytes in memory, instead of 2^{16} bytes.

A.4 Matching Over 7 Rounds

Due to larger size bicliques, even less bytes need to be recomputed. We need to recompute 9 S-boxes in the forward direction and 41 S-boxes in the backward direction, which only gives us 50 in total. Refer to Fig. 6 and Fig. 7 for details.

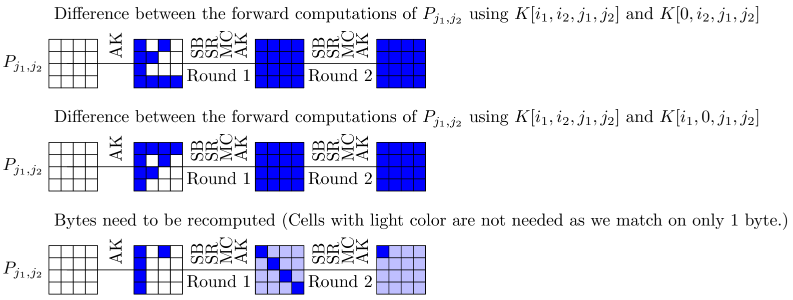

Difference between the forward computations of P_{j_1,j_2} using $K[i_1,i_2,j_1,j_2]$ and $K[0,i_2,j_1,j_2]$
P_{j_1,j_2}
AK
SB SR MC AK
Round 1
Round 2
Difference between the forward computations of P_{j_1,j_2} using $K[i_1,i_2,j_1,j_2]$ and $K[i_1,0,j_1,j_2]$
Bytes need to be recomputed (Cells with light color are not needed as we match on only 1 byte.)

Fig. 6. Forward recomputation in matching: AES-128

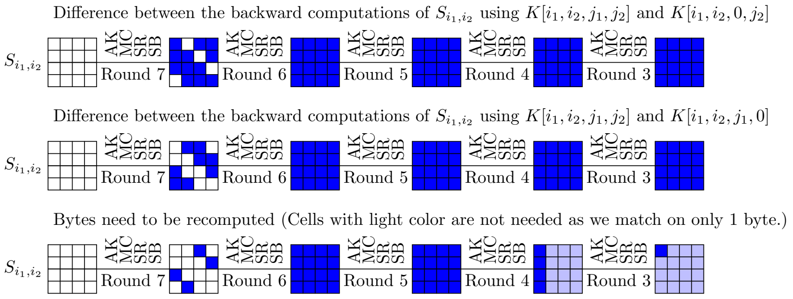

Difference between the backward computations of S_{i_1,i_2} using $K[i_1,i_2,j_1,j_2]$ and $K[i_1,i_2,0,j_2]$
S_{i_1,i_2}
AK MC SR SB
Round 7
Round 6
Round 5
Round 4
Round 3
Difference between the backward computations of S_{i_1,i_2} using $K[i_1,i_2,j_1,j_2]$ and $K[i_1,i_2,j_1,0]$
Bytes need to be recomputed (Cells with light color are not needed as we match on only 1 byte.)

Fig. 7. Backward recomputation in matching: AES-128

A.5 Complexity of the Attack

The evaluation of time and data complexities is similar to those in Sect. 4.5, and thus are omitted here. Refer to Table 1 for the results.

Memory Complexity. Fig. 8 illustrates the bytes need to be stored in the matching phase, which suggests a total of 23×2^{24} bytes needs to be stored. Coupled with a total of 32×2^{24} bytes in the round key (those $*$ and $\circ$ in Fig. 2), it seems the total memory complexity is more than $23 \times 2^{24} + 32 \times 2^{24} = 2^{29.78}$ bytes. This complexity can be reduced if we fixed the value of j_1 first, and do the precomputation and recomputation with each fixed j_1.

To be specific, for each fixed j_1, we need to stores 7 bytes for each different value i_2, j_2 (the first row in Fig. 8), 4 bytes for each different value i_1, j_2 (the second row in Fig. 8) and 8 bytes for each different value i_1, i_2 (the last row in Fig. 8); when j_1 varies to the next value, we store the values of all those bytes again for the updated j_1 value. Thus, the memory complexity used for storing state bytes is

$$M_{\text{states}} = 7 \times 2^{16} + 4 \times 2^{16} + 4 \times 2^{24} + 8 \times 2^{16} = 2^{26.03} \text{ (bytes)}.$$

We can do the same for key bytes storage. According to Fig. 2, for each fixed j_1, we need to stores 23 bytes for each different value i_2, j_2 and 9 bytes for

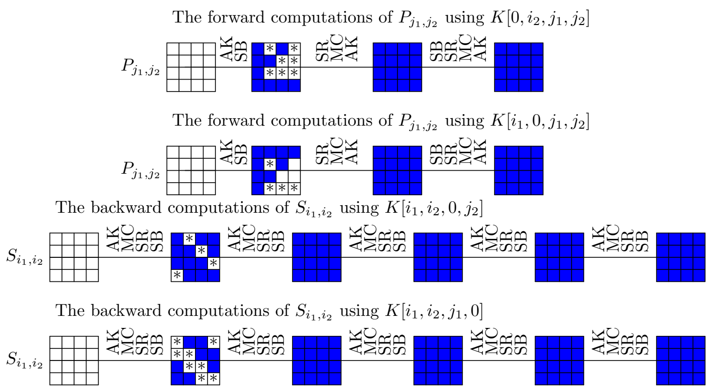

The forward computations of P_{j_1,j_2} using $K[0, i_2, j_1, j_2]$
P_{j_1,j_2}
AK SB
SR MC AK
SB SR MC AK
The forward computations of P_{j_1,j_2} using $K[i_1, 0, j_1, j_2]$
The backward computations of S_{i_1,i_2} using $K[i_1, i_2, 0, j_2]$
S_{i_1,i_2}
AK MC SR SB
The backward computations of S_{i_1,i_2} using $K[i_1, i_2, j_1, 0]$

Fig. 8. Bytes to be stored in states: AES-128

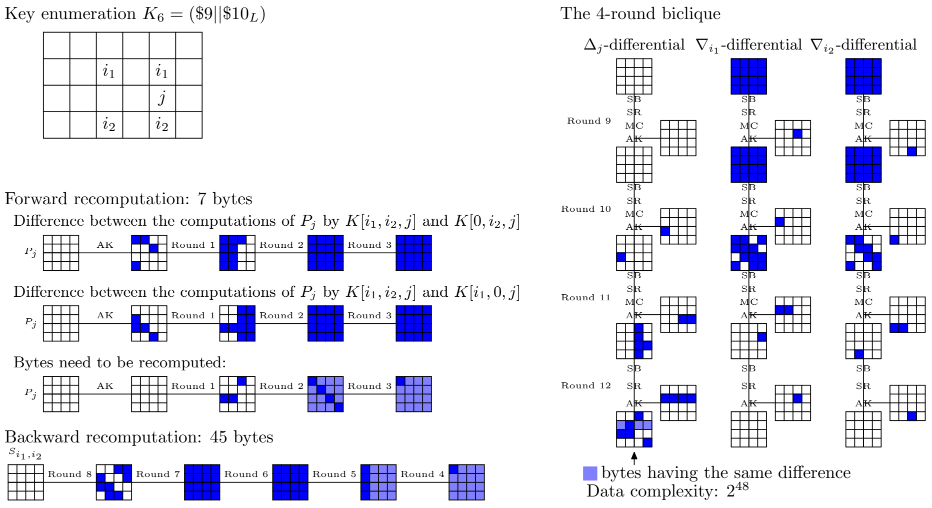

Key enumeration $K_6 = (\$9||\$10_L)$
i_1
j
i_2
Forward recomputation: 7 bytes
Difference between the computations of P_j by $K[i_1, i_2, j]$ and $K[0, i_2, j]$
P_j
AK
Round 1
Round 2
Round 3
Difference between the computations of P_j by $K[i_1, i_2, j]$ and $K[i_1, 0, j]$
Bytes need to be recomputed:
Backward recomputation: 45 bytes
S_{i_1,i_2}
Round 8
Round 7
Round 6
Round 5
Round 4
The 4-round biclique
Δ_j-differential
∇_{i_1}-differential
∇_{i_2}-differential
SB SR MC AK
Round 9
Round 10
Round 11
Round 12
bytes having the same difference
Data complexity: 2^{48}

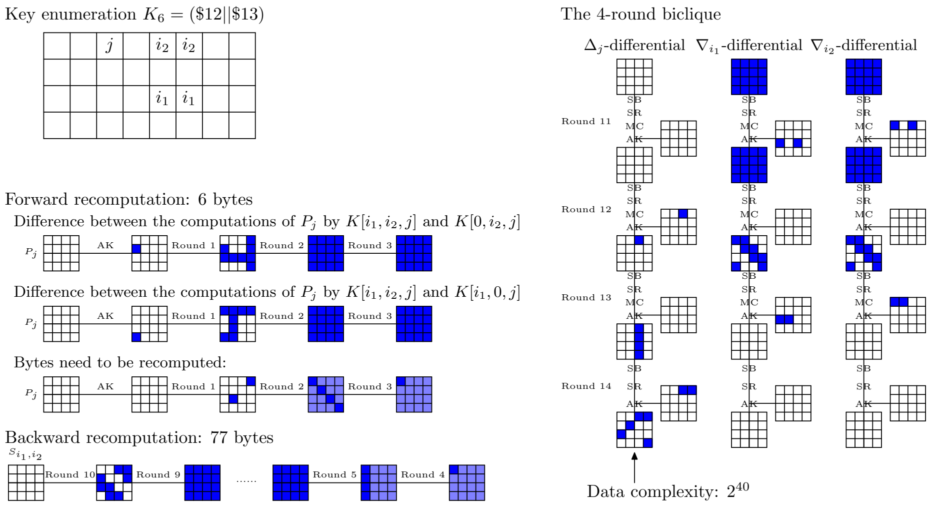
Fig. 9. Illustration for AES-192

each different value i_1, j_2, which has total memory complexity 32×2^{16}. Adding $2 \times 2^{16} \times 16$ bytes memory needed to store a biclique, we have total memory complexity to be

$$2 \times 2^{16} \times 16 + 2^{26.03} + 32 \times 2^{16} = 2^{26.11} \text{ (bytes)}.$$

B The Improved Attacks Against AES-192 and AES-256

The attacks against AES-192 and AES-256 with $2^{16} \times 2^{8}$ bicliques are similar to the one for AES-128 discussed in Sect. 4. Here, we give only figures illustrations Fig. 9 and Fig. 10 showing details of key enumerations, biclique constructions



Key enumeration $K_6 = (\$12||\$13)$
j
i_2
i_2
i_1
i_1
Forward recomputation: 6 bytes
Difference between the computations of P_j by $K[i_1, i_2, j]$ and $K[0, i_2, j]$
P_j
AK
Round 1
Round 2
Round 3
Difference between the computations of P_j by $K[i_1, i_2, j]$ and $K[i_1, 0, j]$
Bytes need to be recomputed:
Backward recomputation: 77 bytes
S_{i_1,i_2}
Round 10
Round 9
Round 5
Round 4
The 4-round biclique
Δ_j-differential
∇_{i_1}-differential
∇_{i_2}-differential
Round 11
Round 12
Round 13
Round 14
SB
SR
MC
AK
Data complexity: 2^{40}

Fig. 10. Illustration for AES-256

and recomputations in both directions. Following Sect. 4, the reader can easily verify the results in Table 1 from these figures.

References

1. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: A framework for automated independent-biclique cryptanalysis. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 561–582. Springer, Heidelberg (2014)
2. Bahrak, B., Aref, M.R.: Impossible differential attack on seven-round AES-128. Information Security, IET **2**(2), 28–32 (2008)
3. Biham, E., Dunkelman, O., Keller, N.: Related-key impossible differential attacks on 8-round AES-192. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 21–33. Springer, Heidelberg (2006)
4. Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., Shamir, A.: Key recovery attacks of practical complexity on AES-256 variants with up to 10 rounds. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 299–319. Springer, Heidelberg (2010)
5. Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 1–18. Springer, Heidelberg (2009)
6. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and related-key attack on the full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
7. Bogdanov, A., Chang, D., Ghosh, M., Sanadhya, S.K.: Bicliques with minimal data and time complexity for AES. In: Lee, J., Kim, J. (eds.) Information Security and Cryptology-ICISC 2014. LNCS, pp. 160–174. Springer, Heidelberg (2015)

8. Bogdanov, A., Kavun, E., Paar, C., Rechberger, C., Yalcin, T.: Better than brute-force–optimized hardware architecture for efficient biclique attacks on AES-128. In: ECRYPT Workshop, SHARCS-Special Purpose Hardware for Attacking Cryptographic Systems (2012)
9. Bogdanov, A., Khovratovich, D., Rechberger, C.: Biclique cryptanalysis of the full AES. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 344–371. Springer, Heidelberg (2011)
10. Canteaut, A., Naya-Plasencia, M., Vayssière, B.: Sieve-in-the-middle: improved MITM attacks. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 222–240. Springer, Heidelberg (2013)
11. Chen, Sz, Xu, Tm: Biclique attack of the full ARIA-256. IACR Cryptology ePrint Archive **2012**, 11 (2012)
12. Çoban, M., Karakoç, F., Boztaş, Ö.: Biclique cryptanalysis of TWINE. In: Pieprzyk, J., Sadeghi, A.-R., Manulis, M. (eds.) CANS 2012. LNCS, vol. 7712, pp. 43–55. Springer, Heidelberg (2012)
13. Daemen, J., Rijmen, V.: The design of Rijndael: AES - the advanced encryption standard. Springer Science & Business Media (2002)
14. Dunkelman, O., Keller, N., Shamir, A.: Improved single-key attacks on 8-round AES-192 and AES-256. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 158–176. Springer, Heidelberg (2010)
15. Gorski, M., Lucks, S.: New related-key boomerang attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 266–278. Springer, Heidelberg (2008)
16. Hong, D., Koo, B., Kwon, D.: Biclique attack on the full HIGHT. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 365–374. Springer, Heidelberg (2012)
17. Jakimoski, G., Desmedt, Y.: Related-key differential cryptanalysis of 192-bit key AES variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
18. Khovratovich, D., Leurent, G., Rechberger, C.: Narrow-bicliques: cryptanalysis of full IDEA. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 392–410. Springer, Heidelberg (2012)
19. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: attacks on skein-512 and the SHA-2 family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
20. Li, L., Jia, K., Wang, X.: Improved single-key attacks on 9-round AES-192/256. In: Cid, C., Rechberger, C. (eds.) FSE 2014. LNCS, vol. 8540, pp. 127–146. Springer, Heidelberg (2015)
21. Mala, H.: Biclique-based cryptanalysis of the block cipher SQUARE. Information Security, IET **8**(3), 207–212 (2014)
22. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: Progress in Cryptology-INDOCRYPT 2010, pp. 282–291. Springer (2010)
23. Wang, Y., Wu, W., Yu, X.: Biclique cryptanalysis of reduced-round piccolo block cipher. In: Ryan, M.D., Smyth, B., Wang, G. (eds.) ISPEC 2012. LNCS, vol. 7232, pp. 337–352. Springer, Heidelberg (2012)