The First Biclique Cryptanalysis of Serpent-256

Gabriel C. de Carvalho[1], **Luis A. B. Kowada**[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF) – Niterói – RJ – Brazil

Abstract. *The Serpent cipher was one of the finalists of the AES process and as of today there is no method for finding the key with fewer attempts than that of an exhaustive search of all possible keys, even when using known or chosen plaintexts for an attack. This work presents the first two biclique attacks for the full-round Serpent-256. The first uses a dimension 4 biclique while the second uses a dimension 8 biclique. The one with lower dimension covers nearly 4 complete rounds of the cipher, which is the reason for the lower time complexity when compared with the other attack (which covers nearly 3 rounds of the cipher). On the other hand, the second attack needs a lot less pairs of plaintexts for it to be done. The attacks require* $2^{255.21}$ *and* $2^{255.45}$ *full computations of Serpent-256 using* 2^{88} *and* 2^{60} *chosen ciphertexts respectively with negligible memory.*

1. Introduction

The Serpent cipher is, along with MARS, RC6, Twofish and Rijindael, one of the AES process finalists [Nechvatal et al. 2001] and has not had, since its proposal, its full round versions attacked. It is a Substitution Permutation Network (SPN) with 32 rounds, 128 bit block size and accepts keys of sizes 128, 192 and 256 bits.

Serpent has been targeted by several cryptanalysis [Kelsey et al. 2000, Biham et al. 2001b, Biham et al. 2001a, Biham et al. 2003, Collard et al. 2007b, Collard et al. 2007a, Nguyen et al. 2011] since the year 2000, but no one was able to get good results on the full cipher, only doing so on reduced round versions of it. These attacks are based on linear crytanalysis [Biham et al. 2001a, Collard et al. 2007b], multiple linear [Collard et al. 2007a], multidimensional linear [Nguyen et al. 2011] and differential-linear cryptanalysis [Biham et al. 2003]. Non-linear attacks are a boomerang attack [Kelsey et al. 2000] and a rectangle attack [Biham et al. 2001b]. The best known attack so far is by Nguyen *et al.* [Nguyen et al. 2011] successfully using multidimensional linear cryptanalysis on 11 and 12-round Serpent as shown in Table 1.

Another famous SPN cipher like the cipher Serpent is Rijindael, the winner of the AES process. Rijindael had its full round versions successfully attacked using Biclique Cryptanalysis in 2011 [Bogdanov et al. 2011] for the first time. Following years brought variations and improvements over the first attack, usually in the form of using better bicliques to achieve lower time or data complexity [Bogdanov et al. 2014, Tao and Wu 2015].

Apart from being used on Rijindael, many other ciphers were successfully attacked, such as HIGHT [Hong et al. 2011], IDEA [Khovratovich et al. 2012], TWINE [Çoban et al. 2012], SQUARE [Mala 2014] and ARIA [Chen and Xu 2014]. However, besides Rijindael, Biclique Cryptanalysis was applied to no other AES finalist.

Attack	Key Size	N^o rounds	Time	Data	Memory
[Kelsey et al. 2000]	192	8	2^{179}	2^{114}	2^{119}
[Biham et al. 2001b]	256	10	$2^{204.4}$	$2^{126.8}$	$2^{131.8}$
[Biham et al. 2001a]	192/256	11	2^{187}	2^{118}	2^{193}
[Collard et al. 2007a]	192	11	2^{178}	2^{118}	2^{88}
[Biham et al. 2003]	192	11	$2^{139.2}$	$2^{125.3}$	2^{60}
[Collard et al. 2007b]	192	11	$2^{116.3}$	2^{118}	2^{108}
[Nguyen et al. 2011]	256	12	$2^{228.8}$	2^{118}	2^{228}
[Nguyen et al. 2011]	256	12	$2^{237.5}$	2^{116}	2^{121}

Table 1. Comparison of time, data and memory complexity between attacks on Serpent, where memory is given in bytes.

Our Contributions. Here we describe two biclique attacks on full round Serpent-256, using balanced bicliques following the basic method by Bogdanov *et al.* [Bogdanov et al. 2011]. It results in the first attacks that are faster than an exhaustive search on full round Serpent as shown in Table 2 below. The first attack is faster but requires more data than the second. Memory used by both is negligible.

Attack	Key Size	N^o rounds	Time	Data	Memory
Second attack	256	32	$2^{255.45}$	2^{60}	$< 2^{19}$
First attack	256	32	$2^{255.21}$	2^{88}	$< 2^{15}$

Table 2. Time, data and memory complexity of our attack, where memory is given in bytes.

Paper structure. Section 2 describes the basics of biclique cryptanalysis while Section 3 describes the Serpent cipher and its notation. Sections 4 and 5 describe the attacks. Finally, Section 6 compares and analyses both attacks and Section 7 concludes the article.

2. Biclique Cryptanalysis

The attack used here, as presented by Bogdanov *et al.* [Bogdanov et al. 2011], is divided into the preparation phase and three steps:

- **Preparation phase.** An adversary partitions the key space into groups with 2^{2d} keys for some d. Each key group is associated with a $2^d \times 2^d$ matrix K, where each element $K[i, j]$ represents a key in the group. Let k be the number of bits of the secret key. In this case we have 2^{k-2d} groups. Also, the cipher $E = f \circ g$ being attacked is a composition of two subciphers f and g. The three steps below are then done for each key group.

1. **Building the biclique.** A biclique structure is built either over the subcipher f or g. In our case it is constructed over f, resulting in a structure that satisfies the condition
$$\forall i, j : S_j \xrightarrow[f]{K[i,j]} C_i,$$
where $0 \le i, j < 2^d$, S_j are internal states of the cipher and C_i are ciphertexts. Section 2.1 explains how it is built.

2. **Obtain plaintexts.** Since this is a chosen ciphertext attack, we have at our disposal a decryption oracle, which is used to obtain the plaintext P_i for each ciphertext C_i.
$$\forall i : C_i \xrightarrow[E^{-1}]{\textit{decryption oracle}} P_i.$$
3. **Meet-in-the-Middle.** For each key $K[i, j]$ in the group it is tested if
$$\exists i, j : P_i \xrightarrow[g]{K[i,j]} S_j.$$
If one of the $K[i, j]$ is the secret key, then the above condition is satisfied. Therefore, every key that satisfies it is a candidate to the secret key. Section 2.2 shows a way, created by Bogdanov *et al.* [Bogdanov et al. 2011], to do this faster than a simple meet-in-the-middle approach.

2.1. Bicliques

First presented by Bogdanov *et al.* [Bogdanov et al. 2011], we now look at the biclique structure. Let f be the subcipher that maps an internal state S to the ciphertext C using the key K (*i.e.* $f_K(S) = C$). A *dimension d biclique over f* is the 3-tuple $(\{S_j\}, \{C_i\}, \{K[i, j]\})$, where $0 \leq i, j < 2^d$ and
$$\forall i, j : f_{K[i,j]}(S_j) = C_i.$$

One way to achieve this condition is using related-key differentials. It is important to highlight the fact that this is a single key attack. The related-key model is used only within the key groups.

Let $K[0, 0]$ be the *base key*, *i.e.* the key that maps the internal state S_0 to the ciphertext C_0. This is called the *base computation*
$$S_0 \xrightarrow[f]{K[0,0]} C_0.$$

The next step is defining the Δ_i*-differentials* and ∇_j*-differentials* using related-key differentials.

Δ_i*-differentials* map the input difference 0 to the output difference Δ_i, using the key difference Δ_i^K, where $\Delta_0 = \Delta_0^K = 0$ and $0 \leq i < 2^d$
$$0 \xrightarrow[f]{\Delta_i^K} \Delta_i,\ \Delta_0 = \Delta_0^K = 0.$$

In contrast, ∇_j*-differentials* map the input difference ∇_j to the output difference 0, using the key difference ∇_j^K, where $\nabla_0 = \nabla_0^K = 0$ and $0 \leq j < 2^d$
$$\nabla_j \xrightarrow[f]{\nabla_j^K} 0,\ \nabla_0 = \nabla_0^K = 0.$$

If both sets of differentials are independent (do not share non-linear components, such as S-boxes), then it is possible to combine them into (Δ_i, ∇_j)*-differentials*
$$\nabla_j \xrightarrow[f]{\nabla_j^K \oplus \Delta_i^K} \Delta_i.$$

Being independent means that an internal state or subkey of f is affected by ∇_j-*differentials* if and only if it is not affected by Δ_i-*differentials.*

By definition, the base computation conforms to both sets of differentials and hence, it is possible to substitute it to the combined differentials

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

By letting

$$S_j = S_0 \oplus \nabla_j,$$
$$C_i = \Delta_i \oplus C_0 \text{ and}$$
$$K[i,j] = K[0,0] \oplus \nabla_j^K \oplus \Delta_i^K$$

we have the definition of a dimension d biclique over f.

Building a biclique this way costs only 2^{d+1} computations of f, since it is possible to choose the key differences and base computation, and then, independently, compute the Δ_i-*differentials* and ∇_j-*differentials.*

2.2. Matching with Precomputations

This technique [Bogdanov et al. 2011] uses the knowledge that only parts of the cipher are affected by the differentials of the biclique to do the meet-in-the-middle faster.

This can be further exploited if instead of meeting an entire internal state, we meet in only a part of the state, namely v. This way we only look at the parts *affected by* the differentials *and* that *affect* v.

Let $E = f \circ s \circ t$, where the biclique was built over f. An adversary then computes and stores 2.2^d *full computations* of the cipher up to the variable v: 2^d computations of t and 2^d computations of s^{-1}

$$\forall i : P_i \xrightarrow[t]{K[i,0]} v_i^1 \text{ and } \forall j : v_j^2 \xrightarrow[s^{-1}]{K[0,j]} S_j.$$

This means that every internal state and subkeys of s and t up to v have to be stored. This is the *precomputation phase*.

Then comes the *recomputation phase*, where the parts that differ from the stored values must be recomputed. The cost of this method is rather variable depending on the diffusion properties of the cipher of interest, both the diffusion related to the key schedule or the internal states. The number of rounds may also influence the cost.

2.3. Complexities

This attack can be seen as an improved exhaustive search, since every key will be tested, but not the whole cipher will be computed in each step. Three types of complexities are of interest: memory, data and time.

The memory complexity is dominated by the Precomputation Phase due to requiring the storage of whole states of many rounds of the cipher. So if the biclique has dimension d, the memory complexity will be 2^{d+1} computations of g.

The data complexity depends only on how many bits of C_i are affected by the Δ_i-differentials, which depends essentially on the amount of rounds covered by the cipher as well as on the dimension and diffusion properties of the cipher.

Finally, the time complexity is where most of the analysis is necessary. It is basically the number of key groups times the time complexity of each iteration. each iteration builds the biclique and then does the matching with precomputations, which is divided into precomputation phase and recomputation phase. In the end, we have

$$C_{time} = 2^{k-2d}(C_{biclique} + C_{precomp} + C_{recomp} + C_{falspos}).$$

The false positives are the keys that pass on the test in the recomputation phase, meaning that they are secret key candidates. Thus it is necessary to check if they are the secret key.

3. The Serpent Cipher

Serpent is a 32 round SPN Cipher. Each internal state and each subkey has 128 bits, divided into 4 words of 32 bits each, while the secret key may have 128, 192 or 256 bits [Biham et al. 1998]. We call Serpent-128 the version of this cipher with 128 bits and similarly for the other key sizes.

Each round i of the cipher consists of 3 layers: key addition called AK_i, substitution phase SB_i and linear transformation L. The exception is the last round in which the linear transform is substituted with another key addition. Rounds vary from 0 to 31 and we have 33 subkeys, derived from the secret key, indexed from K^0 to K^{32}.

There are also two permutations: one *Initial* applied before the first round and one *Final* applied after the last one. However, they do not add any aspect of security to the cipher and thus, can be ignored for the purposes of cryptanalysis.

We call internal state any state of the original plaintext after some operation in the cipher. Therefore, let each internal state be represented by $\#j$, where $0 \leq j \leq 96$, $\#0 = P$ e $\#96 = C$. Each state $\#j$ can be graphically represented by a 4×8 matrix of squares, in which each square represents a *nibble* and each line represents a 32 bit word. The nibbles are enumerated from the leftmost to the rightmost and then down to the next word. The words are enumerated from top to bottom. The h-th nibble of the state S é denoted as S_h. Next we have the graphic representation of an internal state of the cipher.

W_0	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
W_1	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}
W_2	h_{16}	h_{17}	h_{18}	h_{19}	h_{20}	h_{21}	h_{22}	h_{23}
W_3	h_{24}	h_{25}	h_{26}	h_{27}	h_{28}	h_{29}	h_{30}	h_{31}

The substitution phase SB_i looks up to one of 32 different S-boxes for each word, using the same S-box for each nibble, depending on the round. The i-th round uses the S-box B_i. For the purpose of not wasting space, we invite the reader to the Serpent's original paper [Biham et al. 1998] for the description of the S-boxes.

The linear transformation L is the application of a series of rotations ($<<<$), shifts ($<<$) and XORs ($\oplus$) between the words of the current internal state. Let W_0, W_1, W_2 and W_3 be the 4 words of a state S. The following operations are done sequentially:

1. $W_0 <<< 13$. The word W_0 is rotated 13 bits to the left.

2. $W_2 = W_2 <<< 3$. The word W_2 is rotated 3 bits to the left.
3. $W_1 = W_1 \oplus W_0 \oplus W_2$. The XOR operation is applied to W_1, W_0 and W_2.
4. $W_3 = W_3 \oplus W_2 \oplus (W_0 << 3)$. The XOR operation is applied to W_3, W_2 and a 3 bits to the left shifted W_2.
5. $W_1 = W_1 <<< 1$. The word W_1 is rotated 1 bits to the left.
6. $W_3 = W_3 <<< 7$. The word W_3 is rotated 7 bits to the left.
7. $W_0 = W_0 \oplus W_1 \oplus W_3$. The XOR operation is applied to W_0, W_1 and W_3.
8. $W_2 = W_2 \oplus W_3 \oplus (W_1 << 7)$. The XOR operation is applied toW_2, W_3 and a 7 bits to the left shifted W_1.
9. $W_0 = W_0 <<< 5$. The word W_0 is rotated 5 bits to the left.
10. $W_2 = W_2 <<< 22$. The word W_2 is rotated 22 bits to the left.

Finally, we have the key schedule. It receives the 128, 192 or 256-bit secret key as input (depending on the version of the cipher) and generates 33 subkeys with 128 bits each. Independently from the version, the input has 256 bits. So, for the smaller keys, the following bit is set to 1 and the rest is set to 0. For instance, in the Serpent-128, the 128th bit is set to 1 and every bit from 129 to 255 is set to 0. The input has 8 words indexed from w_{-8} to w_{-1}. Then, the *pre-key* is calculated, which are 132 words indexed from w_0 to w_{131} in the following manner:

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus i \oplus \phi) << 11 \qquad (1)$$

where ϕ is the fractional part of the golden ratio $(\sqrt{5}+1)/2$ or $0x9e3779b9$ in hexadecimal.

From the pre-key we generate the 133 words of the subkeys. Each word can be written as $k_i = SBox_{(3-(i \mod 33)) \mod 32}(w_i)$. For example, $k_{100} = SBox_2(w_{100})$. Lastly, the subkey K^i is formed by the words $k_{4i}, k_{4i+1}, k_{4i+2}$ and k_{4i+3}, and $0 \leq i \leq 32$.

More detailed information about the design can be obtained in the original proposal [Biham et al. 1998].

4. First Attack: Dimension 4 Biclique

4.1. Key Partitioning

The first step for the attack is the key partitioning. We partition the key into 2^{k-2d} groups where k is the number of bits in the secret key and d is the dimension of the biclique. Since this attack uses a dimension 4 biclique and is applied to the Serpent-256, there are 2^{248} groups. Each group is represented by a base key.

A base key defines a 2^8-key group and each iteration tests one of these groups. Therefore we chose the subkeys K^{29} and K^{30} to define those groups. This is possible because two sequential subkeys are capable of generating all others. Nibbles K^{30}_{20} and K^{30}_{31} are set to zero while all others of the subkeys K^{29} and K^{30} will vary from iteration to iteration, generating a new base key to test a new group.

The base key of a group is called $K[0,0]$ and the key $K[i,j] = K[0,0] \oplus \Delta^K_i \oplus \nabla^K_j$ belongs to this group.

4.2. 4 Round Dimension 4 Biclique

This biclique covers the last 4 rounds of the cipher excluding the operation AK_{28}, *i.e.* from $\#85$ to $\#96$. In other words, it is 4 rounds minus one application of L.

First, an adversary sets $C_0 = 0$ and calculates $S_0 = f^{-1}_{K[0,0]}(C_0)$, where $K[0,0]$ is the base key of the group. Then, the Δ_i-differentials are computed using the key differentials Δ_i^K on the subkeys K^{29} and K^{30}. The ∇_j-differentials are computed using the key differentials ∇_j^K on the subkeys K^{31} and K^{32}.

The key differentials were chosen in such a way that the Δ_i-differentials and ∇_j-differentials are independent.

Figure 1 shows the basic structure for building our dimension 4 biclique with 2^5 computations of f. It is a dimension 4 biclique because the differentials share no non-linear components (*i.e.* S-boxes) and there are 2^4 possible C_i and S_j.

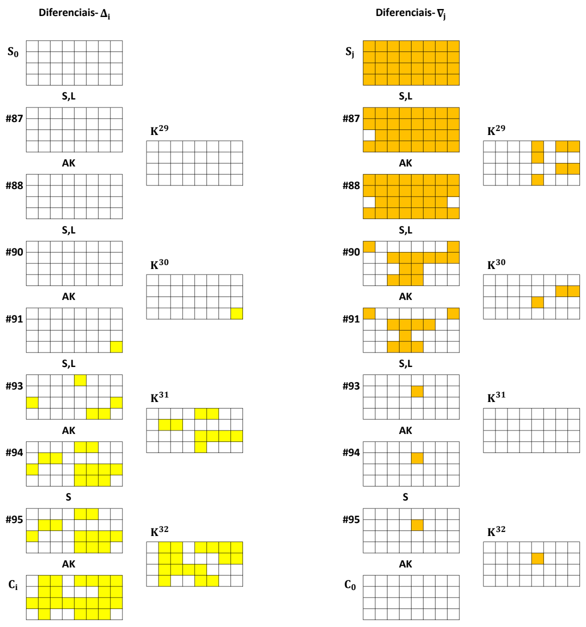

Figure 1. Δ_i-differentials and ∇_j-differentials in the dimension 4 biclique

It is possible to notice that only 22 of C_i are affected and thus, only 2^{88} pairs of plaintexts/ciphertexts are necessary for the attack, *i.e.* there are only 2^{88} possibilities for the ciphertexts.

4.3. Matching with Precomputations over 28 rounds

In this part of the attack, we check if the secret key belongs to the group $\{K[i,j]\}$, *i.e.* if $K_{secret} \in \{K[i,j]\}$. First we precompute 2^5 values of v, which we define as being the nibbles $\#6_6$ and $\#6_7$, and save them, together with all internal states and subkeys involved in these precomputations. Then, we have

$$P_i \xrightarrow[h]{K[i,j]} v^1_{i,j} \text{ and } v^2_{i,j} \xleftarrow[g]{K[0,j]} S_j$$

for each i and j, recomputing only those parts that differ from the ones saved in memory. If $v^1_{i,j} = v^2_{i,j}$, then $K[i,j]$ is a key candidate.

4.3.1. Forward Recomputation

Here we observe the difference between the computation of $P_i \xrightarrow{K[i,j]} v$ and the precomputed values of $P_i \xrightarrow{K[i,0]} v$, given by the influence of ∇_j^K on the subkeys K^0 and K^1.

Only 18 nibbles of K^0 are influenced by ∇_j^K, but three of them do not affect v thus, they can be ignored. Although K^1 has 20 of its nibbles influenced by ∇_j^K, only three of them affect v. Figure 2(a) shows the nibbles that affect the computation of v. Therefore, we only have to recompute the S-boxes 24 times for the internal states and 18 nibbles of the keys K^0 and K^1. The total is 42 S-boxes recomputations.

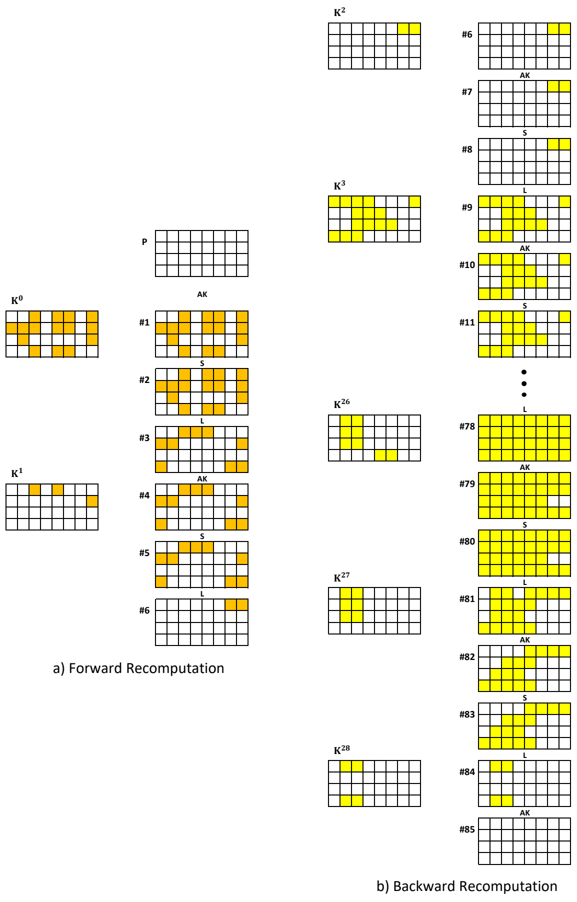

Figure 2. (a) Forward recomputations and (b) backward recomputations of the dimension 4 biclique

4.3.2. Backward Recomputation

Similarly to the forward recomputation, we look at the difference between $v \xleftarrow{K[i,j]} S_j$ and the precomputed $v \xleftarrow{K[0,j]} S_j$, given by the influence of Δ_i^K in the subkeys between K^2 and K^{28}.

Figure 2(b) shows that K^{28} affects 4 nibbles of #84, which implies that 15 S-box computations on state #83. After K^{27}, 30 computations are necessary on state #80.

From $\#77$ to $\#10$, all Substitution Phases must be recomputed. However, on the internal state $\#11$, only 15 nibbles affect v, and thus, only 15 S-boxes must be recomputed. The same amount is valid for the subkey K^3.

Finally, only 2 more computations are necessary for $\#6$ and K^2 each, totaling 766 S-boxes for all internal states and 386 for the subkeys.

4.4. Complexities

Firstly we have

$$C_{total} = 2^{k-2d}(C_{biclique} + C_{precomp} + C_{recomp} + C_{falpos}).$$

We know that there will be only one false positive on average per iteration ($C_{falpos} = 2^{2d}/2^{|v|}$). We also know that $C_{biclique} = 2^5 \cdot (4/32) = 2^2$ and $C_{precomp} = 2^4 \cdot (28/32) = 2^{3.81}$. It remains to find C_{recomp}. For that, we shall count each S-box computation done in the recomputation phase and compare the total number of S-box computations involved in the computation of the whole cipher. Therefore, we know which percentage of the cipher is computed in each iteration, observing only the number of S-boxes, due to the fact that this is the most expensive operation of the cipher.

Once it is not necessary to observe the nibbles that do not influence the recomputation (due to the fact that the key schedule only computes the S-boxes after calculating all words of the subkeys), the total of S-box computations done by the key schedule is 390, while the internal states, as shown in Figure 2, do 803 computations, totalling 1194 S-box computations. Since we do not need to recompute the values already calculated on the precomputation phase and the entire cipher computes 2080 S-boxes, then $C_{recomp} = (2^8 - 2^5) \cdot (1194/2080) = 2^{7.01}$.

In the end, we obtain approximately

$$C_{total} = 2^{248}(2^2 + 2^{3.81} + 2^{7.01} + 1) = 2^{255.21}$$

Serpent-256 computations.

In terms of memory, the attack is upper limited by 2^4 computations of g and h, since g and h together are much bigger than f and thus, much more memory is necessary to store all of their internal states and subkeys than storing the 2^5 states necessary for the biclique. The full computation of g and h consists of 85 internal states and 28 subkeys, with 16 bytes each. Therefore the memory complexity is $2^4 \cdot (85 + 28) \cdot 16 = 2^{14.82}$ bytes approximately.

5. Second Attack: Dimension 8 Biclique

5.1. Key Partitioning

In the second attack we have a dimension 8 biclique on the Serpent-256, thus, the 2^{256} possible keys are partitioned in 2^{240} key groups. Therefore, each base key defines a 2^{16} key group.

In this case, we use subkeys K^{30} and K^{31} to define these groups as shown below. The nibbles K^{31}_{11}, K^{31}_{12}, K^{31}_{30} and K^{31}_{31} are set to zero, while the others vary from iteration to iteration.

As in the first attack, all keys are tested because K^{31}_{30} and K^{31}_{31} are directly dependant on the key difference Δ^K_i used in the attack and K^{31}_{11} and K^{31}_{12} are directly dependant on the key difference ∇^K_j. The base key of a group is called $K[0,0]$ and the key $K[i,j] = K[0,0] \oplus \Delta^K_i \oplus \nabla^K_j$ belongs to this group.

5.2. 3 Round Dimension 8 Biclique

This biclique covers the 3 last rounds of the cipher with exception to the addition of the key K^{29}, *i.e.* from state #88 to #96. Basically it is 3 rounds excluding an application of L.

As with the first attack, an adversary sets $C_0 = 0$ and computes $S_0 = f^{-1}_{K[0,0]}(C_0)$, where $K[0,0]$ is the base key of the current key group. Next the Δ_i-differentials are computed based on the key difference Δ^K_i on subkeys K^{30} and K^{31}. The ∇_i-differentials are computed based on the key difference ∇^K_i on subkeys K^{31} and K^{32}.

The key differentials were chosen in such a way that the Δ_i-differentials and ∇_j-differentials are independent, where i^1 refers to the 4 leftmost bits of i and i^2 refers to the other bits. The same is true for j^1 and j^2.

Figure 3 shows the basic structure for building the dimension 8 biclique with 2^9 computations of f. It is a dimension 8 biclique because the differentials share no non-linear components (*i.e.* S-boxes) and there are 2^8 possible C_i and S_j.

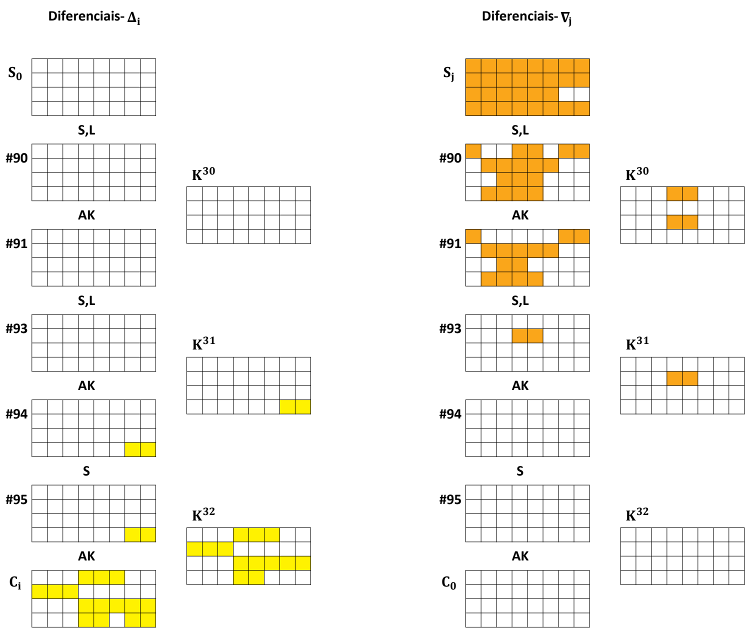

Figure 3. Δ_i-differentials and ∇_j-differentials in the dimension 4 biclique

Through looking at C_i we can notice that 15 nibbles are affected and thus, only 2^{60} plaintext/ciphertext pairs are necessary for the attack since there are only 2^{60} possible ciphertexts.

5.3. Matching with Precomputations over 29 Rounds

Finally we check if the secret key belongs to the group $\{K[i,j]\}$, *i.e.* if $K_{secret} \in \{K[i,j]\}$. We make 2^9 precomputations of v, which we define as being the nibbles, as we did for the first attack, $\#6_6$ and $\#6_7$ and store them together with the internal states and subkeys involved in those precomputations. Then we do:

$$P_i \xrightarrow[h]{K[i,j]} v^1_{i,j} \text{ e}$$

$$v^2_{i,j} \xleftarrow[g]{K[0,j]} S_j$$

for all i and j, recomputing only the necessary parts that differ from the stored ones. If $v^1_{i,j} = v^2_{i,j}$, then $K[i,j]$ is a key candidate.

5.3.1. Forward Recomputation

Here we observe the difference between the computation of $P_i \xrightarrow{K[i,j]} v$ and the precomputed values of $P_i \xrightarrow{K[i,0]} v$, given by the influence of ∇^K_j on the subkeys K^0 and K^1.

We have that 22 nibbles of K^0 and K^1 are influenced by ∇^K_j, but 4 nibbles of K^0 and 15 nibbles of K^1 do not affect v and thus, can be ignored. Figure 4(a) shows the nibbles of K^1 that affect v's computation. Therefore, we have a total of 25 S-box computations for the keys and 27 more for the internal states, totaling 52 S-box computations.

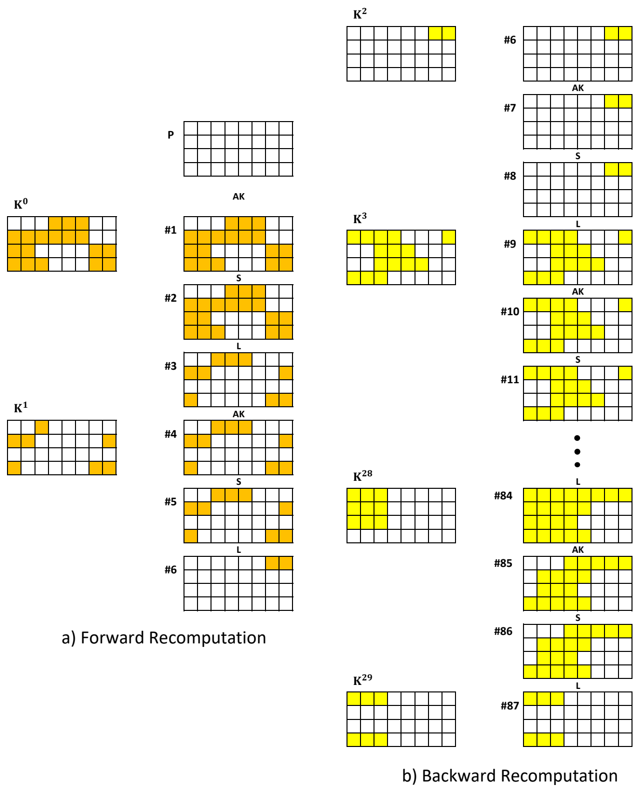

Figure 4. (a) Forward recomputations and (b) backward recomputations of the dimension 8 biclique

5.3.2. Backward Recomputation

Similarly to the forward recomputation, we look at the difference between $v \xleftarrow{K[i,j]} S_j$ and the precomputed $v \xleftarrow{K[0,j]} S_j$, given by the influence of Δ_i^K in the subkeys between K^2 and K^{29}.

Figure 4(b) shows that K^{29} affects 6 nibbles of $\#87$, which then causes in the state $\#86$, after the transformation L, 15 S-box computations. All Substitution Phases must be recomputed from there on, from state $\#83$ to $\#10$. However, in the internal state $\#11$, only 15 nibbles affect v, and thus, only 15 S-box computations are necessary in this state. The same amount is true for subkey K^3.

Lastly, only 2 S-box computations are necessary for $\#8$ and K^2 each, totaling 802 S-boxes for all internal states, and 558 for the subkeys.

5.4. Complexities

Now we use the same formula as in section 4.4. In average we have $C_{falpos} = 2^{16}/2^8 = 2^8$ false positives per iteration and each of them must be tested. We also know that $C_{biclique} = 2^9 \cdot (3/32) = 2^{5.58}$ e $C_{precomp} = 2^8 \cdot (29/32) = 2^{7.86}$. It remains to find C_{recomp} that, as we did before, is calculated by counting each S-box computed in the recomputation phase and comparing it to the total number of S-box computations involved in the computation of the full cipher. Therefore, we will know the percentage of the cipher computed per iteration, observing only the number of S-boxes used, due to it being the most expensive operation of the cipher.

The total of S-box computations made by the key schedule is 583, while the internal states, as shown in Figure 4 make 829 computations, totaling 1412 S-boxes. Since it is not necessary to recompute the precomputed values, and the full cipher makes 2080 S-box computations, then $C_{recomp} = (2^{16} - 2^9) \cdot (1412/2080) = 2^{15.43}$.

Finally, we have

$$C_{total} = 2^{240}(2^{5.58} + 2^{7.86} + 2^{15.43} + 2^8) = 2^{255.45}.$$

Serpent-256 full computations.

In terms of memory, the attack is upper bounded by 2^8 computations of g and h, since together they are much bigger than f and thus, much more memory is necessary to store all of their internal states and subkeys than it is to store the 2^9 states needed for the biclique. The full computation of g and h consists of 87 internal states and 29 subkeys, with 16 bytes each. Therefore the memory complexity is $2^8 \cdot (87 + 29) \cdot 16 = 2^{18.86}$ bytes approximately.

6. Analysis of the Attacks

Biclique Cryptanalysis is a very powerful technique for block ciphers and Serpent is not an exception. The attacks presented here have time complexity $2^{255.21}$ for the dimension 4 and $2^{255.45}$ for the dimension 8 bicliques respectively, for the Serpent-256.

The biclique with smaller dimension has memory complexity equivalent to storing less than 2^4 full computations of the Serpent-256 (which means storing all the 96 internal states and 33 subkeys of each full computation) and uses 2^{88} chosen ciphertexts and their plaintexts. The other biclique's memory complexity is equivalent to storing less than 2^{8} full computations of Serpent-256 and uses 2^{60} chosen ciphertexts and their plaintexts.

Therefore, we can see that both attacks have their advantages, where the dimension 4 biclique produced an attack with less time complexity, while the other biclique used a much smaller amount of chosen ciphertexts.

This is evidence that SPN block ciphers with low diffusion in the key schedule are vulnerable to the Biclique Cryptanalysis since the key for Serpent (a cipher that for twenty years has never suffered an efficient attack for more than half of its rounds) can be searched in less time than an exhaustive search through biclique cryptanalysis.

It is not yet confirmed the viability of this attack to the Serpent-128 and Serpent-192 versions due to being necessary two consecutive subkeys to generate all 33 subkeys used on the cipher, forcing biclique cryptanalysis applied to the last rounds to be close to $2^{255.5}$ computations, which is much bigger than the exhaustive search for these versions. Despite that, building bicliques on the first rounds is still possible since we can use the secret key as base key. The difference will be noticed on the number of rounds that will be covered by the bicliques, which is still a work in progress.

The possibility of existing better bicliques than the ones studied here is still a possibility, since we are not able to prove that these bicliques are optimal, both in the amount of rounds covered as well as the dimension and time complexity.

7. Concluding Remarks

We presented here two attacks to the full round Serpent-256 using the same method applied to Rijindael [Bogdanov et al. 2011], since both ciphers are similar in many ways. This method made it possible to create the first attacks faster than a simple exhaustive search through all possible keys, although, due to requiring a huge amount of chosen ciphertexts (2^{88} and 2^{60}), this attack is less practical than the simple brute force.

The first attack is a dimension 4 biclique covering the 4 last rounds of the cipher, having time complexity of $2^{255.21}$, data complexity of 2^{88} and memory complexity of $2^{14.82}$. The second one is a dimension 8 biclique covering the 3 last rounds of the cipher with time complexity $2^{255.45}$, data complexity 2^{60} and memory complexity of $2^{18.86}$.

Future work involves the application of variations of the biclique cryptanalysis to all versions of Serpent as well as the search for the best bicliques for this cipher, in order to improve both the time complexity and the data complexity of the attacks.

References

Biham, E., Anderson, R., and Knudsen, L. (1998). Serpent: A new block cipher proposal. In *International Workshop on Fast Software Encryption*, pages 222–238. Springer.

Biham, E., Dunkelman, O., and Keller, N. (2001a). Linear cryptanalysis of reduced round serpent. In *International Workshop on Fast Software Encryption*, pages 16–27. Springer.

Biham, E., Dunkelman, O., and Keller, N. (2001b). The rectangle attack—rectangling the serpent. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 340–357. Springer.

Biham, E., Dunkelman, O., and Keller, N. (2003). Differential-linear cryptanalysis of serpent. In *International Workshop on Fast Software Encryption*, pages 9–21. Springer.

Bogdanov, A., Chang, D., Ghosh, M., and Sanadhya, S. K. (2014). Bicliques with minimal data and time complexity for aes. In *International Conference on Information Security and Cryptology*, pages 160–174. Springer.

Bogdanov, A., Khovratovich, D., and Rechberger, C. (2011). Biclique cryptanalysis of the full AES. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 344–371. Springer.

Chen, S.-z. and Xu, T.-m. (2014). Biclique key recovery for ARIA-256. *IET Information Security*, 8(5):259–264.

Çoban, M., Karakoç, F., and Boztaş, Ö. (2012). Biclique cryptanalysis of TWINE. In *International Conference on Cryptology and Network Security*, pages 43–55. Springer.

Collard, B., Standaert, F.-X., and Quisquater, J.-J. (2007a). Improved and multiple linear cryptanalysis of reduced round serpent. In *International Conference on Information Security and Cryptology*, pages 51–65. Springer.

Collard, B., Standaert, F.-X., and Quisquater, J.-J. (2007b). Improving the time complexity of matsui's linear cryptanalysis. In *International Conference on Information Security and Cryptology*, pages 77–88. Springer.

Hong, D., Koo, B., and Kwon, D. (2011). Biclique attack on the full HIGHT. In *International Conference on Information Security and Cryptology*, pages 365–374. Springer.

Kelsey, J., Kohno, T., and Schneier, B. (2000). Amplified boomerang attacks against reduced-round mars and serpent. In *International Workshop on Fast Software Encryption*, pages 75–93. Springer.

Khovratovich, D., Leurent, G., and Rechberger, C. (2012). Narrow-Bicliques: cryptanalysis of full IDEA. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 392–410. Springer.

Mala, H. (2014). Biclique-based cryptanalysis of the block cipher SQUARE. *IET Information Security*, 8(3):207–212.

Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., and Roback, E. (2001). Report on the development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology*, 106(3):511.

Nguyen, P. H., Wu, H., and Wang, H. (2011). Improving the algorithm 2 in multidimensional linear cryptanalysis. In *Australasian Conference on Information Security and Privacy*, pages 61–74. Springer.

Tao, B. and Wu, H. (2015). Improving the biclique cryptanalysis of AES. In *Australasian Conference on Information Security and Privacy*, pages 39–56. Springer.