Efficient Construction of the Boomerang Connection Table

Orr Dunkelman

Computer Science Department, University of Haifa, Israel

Abstract. Recently, the Boomerang Connection Table was introduced by Cid et al. as a tool to better evaluate the probability of a boomerang distinguisher. To compute the BCT of an n-bit to n-bit S-box, the inventors of the BCT proposed an algorithm that takes $O(2^{3n})$ time. We show that one can construct the same table in only $O(2^{2n})$ time.

1 Introduction

The boomerang attack [6], introduced by Wagner, has become an important tool in cryptanalysis of symmetric-key primitives. Over the years many extensions and small improvements of the boomerang attack were presented, but an essential element in the analysis of the boomerang distinguisher relies on some small technicality — namely, what is the true probability that the boomerang "returns". Examples where the independence assumptions offered too high estimate are presented in [5] whereas methods to increase this "connection" probability are suggested in [1, 2, 4].

Recently, Cid et al. offered a new tool that explains some of the previous results and allows computing the connection probability in a more accurate manner [3]. The *Boomerang Connection Table* suggests the probability that a boomerang of input difference Δ_{IN} "connects" with output difference Δ_{OUT}. Specifically, the entry $(\Delta_{IN}, \Delta_{OUT})$ of the BCT counts the number of inputs x such that:

$$S^{-1}(S(x) \oplus \Delta_{OUT}) \oplus S^{-1}(S(x \oplus \Delta_{IN}) \oplus \Delta_{OUT}) = \Delta_{IN}. \qquad (1)$$

Along the introduction of the BCT, Cid et al. also present an algorithm that takes time $O(2^{3n})$ for an n-bit to n-bit S-box. The algorithm takes any Δ_{IN} and Δ_{OUT} and checks how many values of x satisfy the above relation.

In this short paper we propose a new algorithm that computes the same BCT in time complexity of $O(2^{2n})$.

2 New Algorithm

The new algorithm is based on trying all possible Δ_{IN} values for a given Δ_{OUT} in one "go" of trying all possible values for x. This is based on the following observation — the BCT at entry $(\Delta_{IN}, \Delta_{OUT})$ is determined by the number of pairs x, y satisfying $x \oplus y = \Delta_{IN}$ and $S^{-1}(S(x) \oplus \Delta_{OUT}) \oplus S^{-1}(S(y) \oplus \Delta_{OUT}) = \Delta_{IN}$. Re-arranging these two equations yields that a pair of values x, y contribute a BCT entry if

$$x \oplus y = S^{-1}(S(x) \oplus \Delta_{OUT}) \oplus S^{-1}(S(y) \oplus \Delta_{OUT})$$

which can be re-written to

$$x \oplus S^{-1}(S(x) \oplus \Delta_{OUT}) = y \oplus S^{-1}(S(y) \oplus \Delta_{OUT}). \qquad (2)$$

This last equation is the key element of the improved computation. It is easy to see that the above equation allows computing for every value x the value $x \oplus S^{-1}(S(x) \oplus \Delta_{OUT})$, looking for a collision in these values, where each collision suggests a pair of (x, y) that satisfies Equation (2) (and should increase the value in the BCT entry corresponding to $(x \oplus y, \Delta_{OUT})$.

The resulting algorithm is depicted in Algorithm 1.

Algorithm 1 Our New Algorithm for constructing a BCT

```
 1: for all values of $\Delta_{OUT}$ do
 2:     Initialize an empty table $T$ of $2^n$ lists
 3:     for all values of $x$ do
 4:         Compute $y = x \oplus S^{-1}(S(x) \oplus \Delta_{OUT})$
 5:         Concatenate $x$ to $T[y]$
 6:     end for
 7:     Initialize an array $Column$ of $2^n$ entries
 8:     for all entries in $T$ do
 9:         if the entry is not empty then
10:             for all pairs of values $(x_i, x_j)$ in the entry do
11:                 Increment $Column[x_i \oplus x_j]$ by 1
12:             end for
13:         end if
14:     end for
15:     Print the entries of $Column$ (which correspond to $\Delta_{OUT}$'s column in the
    BCT)
16: end for
```

It is worth noting that one can easily transform the algorithm to print the rows of the BCT by changing Step 4 to computing $x \oplus S(S^{-1}(x) \oplus \Delta_{IN})$.

The analysis is quite straightforward, as Step 1 tries 2^n possible Δ_{OUT}, and the internal loops, Steps 3–6 and Steps 8–14, take $O(2^n)$ time each. The result is an algorithm that takes $O(2^{2n})$ in total for an n-bit S-box.

Acknowledgements

The first author was supported by the Israel Ministry of Science and Technology. The second author was supported by the European Research Council under the ERC starting grant agreement n. 757731 (LightCrypt).

References

1. Biryukov, A., Cannière, C.D., Dellkrantz, G.: Cryptanalysis of SAFER++. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2729, pp. 195–211. Springer (2003)
2. Biryukov, A., Khovratovich, D.: Related-Key Cryptanalysis of the Full AES-192 and AES-256. In: Matsui, M. (ed.) Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5912, pp. 1–18. Springer (2009)
3. Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang Connectivity Table: A New Cryptanalysis Tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer (2018)
4. Dunkelman, O., Keller, N., Shamir, A.: A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony. J. Cryptology 27(4), 824–849 (2014), https://doi.org/10.1007/s00145-013-9154-9
5. Murphy, S.: The Return of the Cryptographic Boomerang. IEEE Trans. Information Theory 57(4), 2517–2521 (2011)
6. Wagner, D.A.: The Boomerang Attack. In: Knudsen, L.R. (ed.) Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer (1999)