The implementation of the Partial Sum Attack on 6-round reduced AES

Francesco Aldà

November 18th 2013

Introduction

The source code of the Partial Sum Attack has been developed as part of my Master Thesis in Mathematics at the University of Trento [Ald13]. The main topic of my work concerned the cryptanalysis of reduced-round versions of the AES. Under the supervision of Prof. Massimiliano Sala[1] and Dr. Riccardo Aragona[2], I worked, in particular, on the implementation and the improvement of one of the most important attacks developed in the last 10 years, the Partial Sum Attack. Even though the code I developed is not fully optimized and its performances can be certainly further improved, we believe that an effective implementation of this attack had not already been done before my work.
In 2001, Ferguson et al. published a paper called *"Improved cryptanalysis of Rijndael"* [FKL+01], where they first described the Partial Sum, a powerful attack on AES reduced to 6 rounds. It is a dedicated attack improving a previous attack described in [DR98]. The latter one exploits the *integral cryptanalysis*, a general technique which is applicable to a big class of SPN block ciphers. This technique was originally designed by Lars Knudsen in the paper presenting the block cipher Square [DKR97], as a dedicated attack against its byte-oriented structure (this is the reason why this class of attacks is commonly known as the *Square Attack*). Subsequently, Stefan Lucks generalized the attack to what he called a *Saturation Attack* and used it to attack Twofish, which is not at all similar to Square, having a Feistel network structure [Luc00].
Since AES inherits many properties from Square, this attack can be easily extended to reduced-round versions of the Advanced Encryption Standard. Remembering that AES operates on a 4×4 matrix of bytes, called the *state*, the Partial Sum Attack and its predecessor, the Square Attack, are both based on the following fact.

[1]Associate Professor in Algebra at the University of Trento
[2]Postdoc in Mathematics at the University of Trento

Fact 1. *Let $b_{i,j}^{(l)}$ be the byte in position (i, j), $i, j \in \{0, 1, 2, 3\}$, of the l^{th} state of a Δ-set (a set of 256 states that are all different in one state byte (active) and all equal in the other state bytes (passive)) after the application of 3 AES rounds. Then the following relation holds:*

$$\sum_{l=1}^{256} b_{i,j}^{(l)} = 0.$$

Exploiting this relation, it is possible to mount a probabilistic attack (the Square Attack) on 4-round AES. In order to extend this attack to a 6-round version, one can add a round at the beginning, using a set of chosen plaintexts in which 4 bytes vary over the 2^{32} possible 4-tuples while the others are fixed. In this way, one obtains 2^{24} different Δ-sets after the first round application. It is also possible to add another round at the end, but the attacker has to do a partial decryption of 2 rounds instead of only one and hence he needs to guess 4 bytes of the final round key and 1 byte of the penultimate round key.

Implementation

In this section, I briefly describe the main ideas used in my implementation. I refer to [FKL$^+$01, Ald13] for further details and explanations.
For clarity, from now on the index l will be always implied and

- with (c_1, c_2, c_3, c_4) I will indicate a particular configuration of four bytes of the ciphertexts;
- with (k_1, k_2, k_3, k_4) I will indicate four bytes of the 6th round key corresponding to the previous configuration;
- with k_5 I will indicate a byte of a column (determined by the configuration chosen for the ciphertexts' bytes) of the 5th round key after the application of MC^{-1};
- finally with x_2, x_3, x_4 I will indicate the three partial sums.

My implementation works as follows:

- one stores the values of (c_1, c_2, c_3, c_4) into a 2^{32} bit-vector. I used this solution because we are working on a field of characteristic 2 and an element gives a nonzero contribute to the summation if and only if it appears an odd number of times. In this way, I could use the minimal amount of RAM resources in order to mount the attack.
- Guessing the value of k_1 and k_2, one can compute $x_2 = f(c_1, c_2, k_1, k_2)$ and store the triple (x_2, c_3, c_4) into a 2^{24} bit-vector.

- Guessing the value of k_3 one can compute $x_3 = g(x_2, c_3, k_3)$ and store (x_3, c_4) into a 2^{16} bit-vector.
- Guessing the value of k_4 one can compute $x_4 = h(x_3, c_4, k_4)$ and store x_4 into a 2^8 bit-vector.
- Guessing the value of k_5 one can compute the final sum and check if the result is 0.

Improvement

In the source code I developed, I also inlaid the improvement of the Partial Sum Attack I introduced in my thesis. In particular, we have been able to take advantage of some hidden information which allows to lower the data amount one needs to obtain in order for the Partial Sum Attack to be successful. I refer to [Ald13] for a formal and exhaustive explanation of this result.

References

[Ald13] F. Aldà, *The Partial Sum Attack on 6-round reduced AES: Implementation and improvement*, 2013.

[DKR97] J. Daemen, L. Knudsen, and V. Rijmen, *The block cipher Square*, Fast Software Encryption, Springer, 1997, pp. 149–165.

[DR98] J. Daemen and V. Rijmen, *AES proposal: Rijndael*, First Advanced Encryption Standard (AES) Conference, 1998.

[FKL$^+$01] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, *Improved cryptanalysis of Rijndael*, Fast software encryption, Springer, 2001, pp. 213–230.

[Luc00] S. Lucks, *The Saturation Attack - a Bait for Twofish*, Cryptology ePrint Archive, Report 2000/046, 2000, `http://eprint.iacr.org/`.