Using Wiedemann's Algorithm to Compute the Immunity Against Algebraic and Fast Algebraic Attacks

Frédéric Didier

Projet CODES, INRIA Rocquencourt, Domaine de Voluceau,
78153 Le Chesnay cedex
frederic.didier@inria.fr

Abstract. We show in this paper how to apply well known methods from sparse linear algebra to the problem of computing the immunity of a Boolean function against algebraic or fast algebraic attacks. For an n-variable Boolean function, this approach gives an algorithm that works for both attacks in $O(n2^nD)$ complexity and $O(n2^n)$ memory. Here $D = \binom{n}{d}$ and d corresponds to the degree of the algebraic system to be solved in the last step of the attacks. For algebraic attacks, our algorithm needs significantly less memory than the algorithm in [ACG+06] with roughly the same time complexity (and it is precisely the memory usage which is the real bottleneck of the last algorithm). For fast algebraic attacks, it does not only improve the memory complexity, it is also the algorithm with the best time complexity known so far for most values of the degree constraints.

Keywords: algebraic attacks, algebraic immunity, fast algebraic attacks, Wiedemann's algorithm.

1 Introduction

Algebraic attacks and fast algebraic attacks have proved to be a powerful class of attacks against stream ciphers [CM03, Cou03, Arm04]. The idea is to set up an algebraic system of equations satisfied by the key bits and to try to solve it. For instance, this kind of approach can be quite effective [CM03] on stream ciphers which consist of a linear pseudo-random generator hidden with non-linear combining functions acting on the outputs of the generator to produce the final output.

For such an attack to work, it is crucial that the combining functions satisfy low degree relations. The reason for this is that it ensures that the algebraic system of equations verified by the secret key is also of small degree, which is in general essential for being able to solve it. This raises the fundamental issue of determining whether or not a given function admits non-trivial low degree relations [MPC04, Car04, DGM04, BP05, DMS05].

For algebraic attacks, in order to find relations of degree at most d satisfied by an n-variable combining Boolean function f, only two algorithmic approaches

R. Barua and T. Lange (Eds.): INDOCRYPT 2006, LNCS 4329, pp. 236–250, 2006.

are known for the time being. The first one relies on Gröbner bases [FA03] and consists of finding minimal degree elements in the polynomial ideal spanned by the ANF of f and the field equations. The second strategy relies on linear algebra, more precisely we can associate to f and d a matrix M such that the elements in the kernel of M give us low degree relations. Building M is in general easy so the issue here is to find non-trivial elements in the kernel of a given matrix or to show that the kernel is trivial.

The linear approach has lead to the best algorithm known so far [ACG+06] that works in $O(D^2)$ where $D = \binom{n}{d}$. There is also the algorithm of [DT06] which performs well in practice and which is more efficient when d is small. Actually, when d is fixed and $n \to \infty$, this last algorithm will be able to prove the non-existence of low degree relation in $O(D)$ for almost all Boolean functions. Note however that if the ANF of f is simple or has a lot of structure, it is possible that the Gröbner basis approach outperforms these algorithms, especially if the number of variables is large (more than 30).

For fast algebraic attacks, only the linear algebra approach has been used. There are now two degree constraints $d < e$ and the best algorithms are the one of [ACG+06] working in $O(ED^2)$ where $E = \binom{n}{e}$ and the one of [BLP06] working in $O(ED^2 + E^2)$.

All these algorithms relying on the linear algebra approach use some refinements of Gaussian elimination in order to find the kernel of a matrix M. Efficiency is achieved using the special structure of M. We will use here a different approach. The idea is that the peculiar structure of M allows for a fast matrix vector product that will lead to efficient methods to compute its kernel. This comes from the following facts:

- There are algorithms for solving linear systems of equations which perform only matrix vector products. Over the finite fields, there is an adaptation of the conjugate gradient and Lanczos algorithm [COS86, Odl84] or the Wiedemann algorithm [Wie86]. These algorithms were developed at the origin for solving large sparse systems of linear equations where one can compute a matrix vector product efficiently. A lot of work has been done on the subject because of important applications in public key cryptography. Actually, these algorithms are crucial in the last step of the best factorization or discrete logarithm algorithms. Notice as well that these algorithms were also used in the context of algebraic attacks against HFE cryptosystems (see [FJ03]).
- Computing a matrix vector product of the matrix involved in the algebraic immunity computation can be done using only the binary Moebius transform. It is an involution which transforms the two main representations of a Boolean function into each other (namely the list of its ANF coefficients and the list of its images).
- The Moebius transform of an n-variable Boolean function can be computed efficiently in $O(n2^n)$ complexity and $O(2^n)$ memory. We will call the corresponding algorithm the fast Moebius transform by analogy with the fast Fourier transform (note that they both rely on the same principle).