Cryptanalysis of Block Ciphers with Overdefined Systems of Equations

Nicolas T. Courtois[1] and Josef Pieprzyk[2]

[1] CP8 Crypto Lab, SchlumbergerSema, 36-38 rue de la Princesse
BP 45, 78430 Louveciennes Cedex, France
http://www.nicolascourtois.net
courtois@minrank.org

[2] ICS, Macquarie University, NSW 2109, Australia
josef@comp.mq.edu.au

Abstract. Several recently proposed ciphers are built with layers of small S-boxes, interconnected by linear key-dependent layers. Their security relies on the fact, that the classical methods of cryptanalysis (e.g. linear or differential attacks) are based on probabilistic characteristics, which makes their security grow exponentially with the number of rounds N_r.

In this paper we study the security of such ciphers under an additional hypothesis: the S-box can be described by an overdefined system of algebraic equations (true with probability 1). We show that this hypothesis is true for both Serpent (due to a small size of S-boxes) and Rijndael (due to unexpected algebraic properties). We study general methods known for solving overdefined systems of equations, such as XL from Eurocrypt'00, and show their inefficiency. Then we introduce a new method called XSL that uses the sparsity of the equations and their specific structure.

The XSL attack has a parameter P, and in theory we show that P should be a constant. The XSL attack would then be polynomial in N_r, with a huge constant that is double-exponential in the size of the S-box. We demonstrated by computer simulations that the XSL attack works well enough on a toy cipher. It seems however that P will rather increase very slowly with N_r. More simulations are needed for bigger ciphers.

Our optimistic evaluation shows that the XSL attack might be able to break Rijndael 256 bits and Serpent for key lengths 192 and 256 bits. However if only P is increased by 2 (respectively 4) the XSL attack on Rijndael (respectively Serpent) would become slower than the exhaustive search. At any rate, it seems that the security of these ciphers **does not grow exponentially** with the number of rounds.

Key Words: block ciphers, AES, Rijndael, Square, Serpent, Camellia, multivariate quadratic equations, MQ problem, overdefined systems of multivariate equations, XL algorithm, Gröbner bases, sparse multivariate polynomials.

Note: This paper is kept on e-print as an archive of the early work, was written between November 2001 and Mai 2002, and is kept unchanged since, except correcting some small errors and typos. This paper contains a general description of the so called first and second XSL attack on block ciphers. A different version, so called compact version of the first XSL attack, is published in Asiacrypt 2002. When studying such attacks, intuition is very tricky, and though Coppersmith and Moh once claimed that they know that such attacks will not work, so far we did not see any serious argument against XSL.

Attacks in 2^{100} on 128-bit AES: This attack, is a simple adaptation of the second XSL attack, exactly as described here, proposed by Murphy and Robshaw. For each S-box of AES, we decompose it as the modified inverse in $GF(256)$ and a multivariate affine function. Then we create 16 variables for this S-box: if x, y are the input and the output of the modified inverse, we will consider $x, x^2, x^4, x^8, x^{16}, x^{32}, x^{64}, x^{128}, y, y^2, y^4, y^8, y^{16}, y^{32}, y^{64}, y^{128}$ as separate variables (and rename them). Then, given all these new variables, the S-boxes will give quadratic equations in these new variables, and all the remaining AES will be described in terms of linear equations. We can then apply the second XSL attack, with $s = 8$, $r = 24$ and $t = 41$. The exact complexity of this attack remains an open problem.

1 Introduction

On October 2nd, 2000, NIST has selected Rijndael as the Advanced Encryption Standard, destined for massive world-wide usage. Serpent was second in the number of votes [1].
In the famous paper from 1949, Claude E. Shannon states that breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type", see [24]. This seemed very easy to achieve so far, as solving systems of equations can become intractable very easily. For example in [8] Ferguson, Shroeppel and Whiting show how to represent Rijndael with one big equation to solve. The equation is so big: 2^{50} terms for a 128-bit cipher, that it has certainly no consequences whatsoever on the security of Rijndael. Similarly, though every cipher can obviously be described in terms of a system of multivariate equations over $GF(2)$, it does not mean that it can be broken. In the last ten years however surprising attacks have appeared in public key cryptography: the cryptanalysis of Matsumoto-Imai cryptosystem [16] by Patarin and the attack on the basic version of HFE cryptosystem by Courtois [6]. In these attacks the security collapses suddenly after discovery (either theoretical or experimental) of the existence of **additional** multivariate equations, that are not obvious and have not been anticipated by the designers of the original cryptosystems. In this paper, the same thing will happen to some block ciphers.
In this paper we reduce the cryptanalysis of Rijndael and Serpent to solving a system of Multivariate Quadratic equations (a.k.a. MQ problem). MQ is not a contrived problem as in [8] and is already known in cryptography. Several public key cryptosystems are based on hardness of MQ, the best of them being probably HFE published at Eurocrypt 1996 [18]. At Crypto'99 and in Eurocrypt'00, Shamir *et al.* showed that though MQ is NP-hard, its complexity drops substantially when the MQ becomes overdefined (more equations than unknowns), see [21, 22] [1]. In this paper we show that if the MQ is sparse and have a regular structure, it becomes still much easier. It turns out that, the systems of quadratic equations obtained for Rijndael and Serpent, will be both overdefined and sparse.
Since the pioneering work of Luby-Rackoff [12], there were many developments on the security of top-level schemes of block ciphers. The state of art in both security proofs and generic attacks for Feistel ciphers can be found in [14] and [17]. However Rijndael is not a Feistel cipher and a more powerful theory has been developed by Vaudenay [25]: it allows to make security proofs against a large class of attacks including linear and differential cryptanalysis, for an arbitrary type of cipher. From this theory Moriai and Vaudenay have developed at Asiacrypt'00 security proofs for idealized versions of several AES candidates [26]. The outcome for Rijndael was somewhat strange: they needed 384 rounds of Rijndael in order to make sure it was secure. Similar results were obtained for Serpent. Therefore it is not completely unsound to believe that some attacks might exist for Rijndael and Serpent, for which the security would grow slowly with the number of rounds. In this paper we present such an attack.
The paper is organized as follows: first we describe a general class of ciphers that includes Rijndael and Serpent. Then we explore algebraic properties of the Rijndael S-box and show that it gives an overdefined system of equations. Such equations will also exist for Serpent for a very different reason. Consequently we write the cryptanalysis of Rijndael and Serpent (and other similar ciphers) as solving an overdefined system of quadratic equations. The general XL attack known for this problem fails and we will present the new attack called XSL that uses the sparsity of the equations (and their structure). It comes in two versions: first is very general, does not use the key schedule, and is studied approximatively in order to investigate the asymptotic behaviour of XSL. The second version does use the key schedule and is designed for concrete cryptanalysis of Rijndael and Serpent, with all the precision necessary. In the Appendix C we present our simulations done on the XSL attack. Finally from the simulation

[1] Remark: The opposite, underdefined case of MQ has been studied in [5].

results and our estimations we will try to apply the XSL attack to Rijndael and Serpent. It will also imply many interesting conclusions about the design of block ciphers.

2 Substitution-Affine Ciphers, Rijndael and Serpent

A natural way to construct cipher is to follow the Shannon's paradigm of mixing confusion layers with diffusion layers [24]. For example SP-networks [7, 10] are combinations of layers of S-boxes with permutations of bits. More generally we may allow linear or affine functions of bits, not only permutations of wires. We call it a SA-cipher.
At Eurocrypt'00 Shamir and Biryukov studied top-level structural attacks against the SA-ciphers, i.e. the attacks do not depend on particular S-boxes used [20]. In our attacks we will use some special properties of the S-boxes.
In this paper we will specify a restricted class of SA-ciphers called XSL-ciphers. Though our attacks are designed for XSL-ciphers, it is obvious that they can be easily extended to all SA-ciphers, and even to other block ciphers (including Feistel ciphers), provided that they use "bad" S-boxes and have a regular structure.

2.1 XSL-ciphers

By definition, an XSL-cipher is a composition of N_r similar rounds:

X The first round $i = 1$ starts with a XOR with the session key K_{i-1}.
S Then we apply a layer of B bijective S-boxes in parallel, each on s bits,
L Then we apply a linear diffusion layer,
X Then we XOR with another session key K_i.
Then if $i = N_r$ we finish, otherwise we increment i and go back to step S.

We denote the key bits used in an XSL-cipher by the variables $K_{i\ j}$ with $i = 0..N_r$ and $j = 1..s * B$. There are $N_r + 1$ session keys, K_0 is the first and K_{N_r} is the last. The number of key bits before expansion is H_K, the number of key bits after expansion is Ek, and the number of bits that are linearly independent among those is L_k. If we pick some L_k key variables $K_{i\ j}$ to form a basis, we will denote by $[K_{i\ j}]$ a linear expression of (any) key bit as a linear combination of the $K_{i\ j}$ that are in the basis.
We call $X_{i\ j}$ the j-th bit of the input of $i - th$ round function of a XSL-cipher, i.e. taken **after** the XOR with the session key. We denote by $Y_{i\ j}$ the j-th bit of the input of the linear part of $i - th$ round function of a XSL-cipher, i.e. taken after the application of the corresponding S-box to the s corresponding $X_{i\ j}$.
Similarly we denote by $Z_{i\ j}$ the j-th bit of the output of the round function (before the XOR with the next session key). In consequence we denote the plaintext by Z_0 and the ciphertext by X_{N_r+1}, however these are constants, not variables.
With these notations $X_{i+1\ j} = Z_{i\ j} \oplus K_{i\ j}$ for all $i = 0..N_r$.

2.2 The Top-level Structure of Rijndael

Rijndael specified in [4], is a special type of XSL-cipher with $s = 8$, $B = 4 * Nb$. We don't give a full description of it, but will recall all the essential facts when necessary. Rijndael has $N_r = 10..14$ rounds. The data in Rijndael is represented as rectangular "states" that are composed of Nb columns, each having the size of 4 S-boxes ($4 * s = 32$ bits). We have either $Nb = 4$, 6 or 8, which gives block sizes of respectively $Nb * 32 = 128$, 192 and 256 bits. The encryption in Rijndael is performed as follows:

X We XOR the session key K_{i-1}.

- S Then we have $B = Nb * 4$ S-boxes on $s = 8$ bits each.
- L Then we have a permutation of bytes called ShiftRow, followed by a linear transformation $GF(256)^4 \rightarrow GF(256)^4$ called MixColumn applied in parallel for each of Nb columns. If $i = N_r$ (in the last round) the MixColumn is omitted.
- X Then we XOR with another session key K_i and either finish, either go to S and continue with another round...

The (unexpanded) key length is $H_k = Nk * 32$ bits with $Nk = 4$, 6 or 8, which is expanded to $Ek = (N_r + 1) * s * B = (N_r + 1) * Nb * 32$ bits.

2.3 The Top-level Structure of Serpent

Serpent described in [1] is an XSL-cipher with $s = 4$, $B = 32$, $N_r = 32$. The block size is always 128 bits. The key length can be $H_k = 128$, 192 or 256 bits, and is also expanded to $Ek = (N_r + 1) * s * B = 1056$ bits.

3 S-boxes and Overdefined Algebraic Equations

The only non-linear part of XSL-ciphers are the S-boxes. Let $F : GF(2)^s \rightarrow GF(2)^s$ be such an S-box $F : x = (x_1..x_s) \mapsto y = (y_1..y_s)$. In Rijndael and Serpent, like for all other "good" block ciphers, the S-boxes are build with "good" boolean functions. There are many criteria on boolean functions that are more or less applied in cryptography. One of them is that each y_i should have a high algebraic degree when expressed as a multivariate polynomial in the x_i. However all this does not assure that there is no "implicit" multivariate equations of the form $P(x_1, \ldots, x_s, y_1, \ldots, y_s)$ that are of low algebraic degree. We will show that for Rijndael, and for Serpent, for very different reasons, a great number of such equations exist.
Such "implicit" equations has already been used to cryptanalyse the Matsumoto-Imai cryptosystem in [16] and the HFE cryptosystem in [6], but apparently it is the first time they will be used in cryptanalysis of block ciphers.
For a specific degree of the equations d (usually $d = 2$) we are interested in the actual number r of such equations $P(x_1, \ldots, x_s, y_1, \ldots, y_s)$. Unlike for "explicit" equations $y_i = f(x_1, .., x_s)$, this number r can be bigger than s. We are also interested in the number of monomials that appear in these equations denoted by t, and counted including the constant term. In general $t \approx \binom{s}{d}$. If $t \ll \binom{s}{d}$, we say that the equations are **sparse**.
If $r = s$, such equations are (approximatively) sufficient to fully describe the S-box: for each y there will be on average 1 solution x. Thus when $r \gg s$, we will say that the system is **overdefined**.

3.1 The quality of S-boxes and Random S-boxes

When r is close to t, we may eliminate most of the terms by linear elimination, and obtain simpler equations that are sparse and maybe even linear. For this reason it is possible to mesure the quality of our system of equations by the ratio $t/r \geq 1$. If t/r is close to 1, the S-box is considered as "bad". From this point of view, both overdefined systems (big r) and sparse systems (small t) will be "bad". Otherwise, if the system is not overdefined and not sparse, $t/r \approx \mathcal{O}(s^{d-1})$, and such an S-box will be "good" (unless s is very small).We will see that the actual contribution of the S-boxes to the complexity of the attacks described in this paper is approximatively $\Gamma = (t/s)^{\lceil t/r \rceil}$. It is possible to show that for a random S-box, the smallest value of Γ that can be achieved will be double-exponential in s, however this can still be relatively small if s is very small, e.g. 4 bits. For different reasons, for both Rijndael and Serpent S-boxes, we will find overdefined systems of equations with quite a small Γ.

3.2 Overdefined Equations on the Serpent S-box

We show that 4-bit S-boxes always do give an overdefined system of multivariate equations. For this we write a 16×37 matrix containing in each row the values of the $t = 37$ monomials $\{1, x_1, .., x_4, y_1, .., y_4, x_1x_2, .., x_1y_1, .., y_3y_4\}$ for each of the $2^s = 16$ possible entries $x = (x_1, .., x_4)$. The rank of this matrix is at most 16, therefore whatever is the S-box, there will be at least $r \geq 37 - 16 = 21$ quadratic equations. This is a very overdefined system since $21 \gg 4$. We have $t/r \approx 1.75$ and $\Gamma = (t/s)^{\lceil t/r \rceil} \approx 86 \approx 2^6$.
We note that a smaller t/r would be achieved with cubic equations on this S-box, but Γ would be much bigger then. It is also possible to consider bi-affine equations. In this case we have $t = 25$ and $r \geq 25 - 16 = 9$ which is still overdefined, however it gives a larger $\Gamma \approx 244 \approx 2^8$.

3.3 Overdefined Equations on the Rijndael S-box

For Rijndael we have $s = 8$. It is quite big compared to Serpent: there are $(2^8)! \approx 2^{1684}$ bijective S-boxes on 8 bits, compared with only $(2^4)! \approx 2^{44}$ for $s = 4$. For this reason we don't expect any useful properties to happen by chance. For example it is easy to see that with the method described above in 3.2 a random S-box on 8 bits will give $r = 0$ because $2^s = 256$ is bigger than the number 137 of possible quadratic terms. Still the Rijndael S-box has been chosen for optimality results with regard to linear, differential and high-order differential attacks, and is currently the unique S-box known that achieves all these optima, see [2, 15] for details. This uniqueness implies many very special properties.
Rijndael S-box is a composition of the "patched" inverse in GF(256) with 0 mapped on itself, with a multivariate affine transformation $GF(2)^8 \to GF(2)^8$. Following [4] we call these functions respectively g and f and we call $S = f \circ g$. Let x be an input value and $y = g(x)$ the corresponding output value. We also note $z = S(x) = f(g(x)) = f(y)$. According to the definition of the S-box:

$$\forall x \neq 0 \qquad 1 = xy$$

This equation gives in turn 8 multivariate bi-linear equations in 8 variables and this leads to 8 bi-affine equations between the x_i and the z_j. As we explain more in details in the Appendix A, 7 of these equations are true with probability 1, and the 8th is true with probability 255/256. The existence of these equations for g and S is obvious. Surprisingly, much more such equations exist. For example we have:

$$x = y * x^2$$

Since $x \mapsto x^2$ is linear, if written as a set of 8 multivariate functions, the above equation gives 8 bi-affine equations between the x_i and the y_j, and in turn between the x_i and the z_j. Moreover this equation in GF(256) is symmetric with respect to the exchange of x and y. Thus we get 16 bi-affine equations true with probability 1 between the x_i and the z_j.
From the above we have 23 quadratic equations between x_i and the z_j that are true with probability 1. We have explicitly computed these equations (see Appendix A), have verified that they are all linearly independent, and have also verified that there are no more such equations (however there would be more if we allowed additional terms, see Appendix A.1). The terms present in these equations are $t = 81$: these are $\{1, x_1, .., x_8, z_1, .., z_8, x_1z_1, .., x_8z_8\}$, there is no terms in x_ix_j or z_iz_j. Here we get $t/r \approx 3.52$ and $\Gamma \approx 2^{13.4}$ (more than for Serpent).

Additional equations for Rijndael We observe that in Rijndael S-box, if x is always different than 0, there 24 linearly independent quadratic equations. For one S-box, the probability of this 24th equation to be true is 255/256. We are interested in probability that it is true for **all** S-boxes in the execution of Rijndael (i.e. we have $x \neq 0$ everywhere). As it has been already pointed out by the authors of [8], this probability is quite big. It is about[2]:

$$(255/256)^{4*Nb*N_r+4*(1+1_{Nk>6})*N_r}$$

This gives between 1/2 for the smallest Rijndael 128 bits and about 1/9 for the biggest 256-bit version. Therefore if an attack works better with 24 equations, it will usually be worthwhile to use them all and repeat the whole attack 2-9 times. For this reason, if an attack uses only one (or two) executions of the cipher we will assume $r = 24$, otherwise we have $r = 23$.

4 The MQ attack on Block Ciphers

It is obvious that for any SA-cipher such that S-boxes can be described in terms of some algebraic equations, the cryptanalysis of the cipher can be written as a problem of solving a system of such equations. If these equations are Multivariate Quadratic, we call this attack "MQ attack". It is the case for Rijndael and Serpent, as shown above in 3.3 and 3.2.

4.1 The Attack Scenarios

There are many ways in which the MQ attack can be applied. The system of equations should be written in such a way that they should have exactly one solution. For this it is sufficient in practice to build a system having one solution on average. Then if there are a few solutions, prior to the solving stage, we would guess and fix a few bits.

First (general) attack ignoring the key schedule This attack is designed for any XSL-cipher, whatever is the key schedule. Since there are $(N_r + 1)$ keys K_i that are of the same size as a plaintext, and we want enough constraints to determine them (about) uniquely, we will need $(N_r + 1)$ known plaintexts. A better version will use a set of chosen plaintexts that differ by only a few bits in one single S-box. Thus we will have many common variables between systems of equations written for different plaintext/ciphertext pairs.
This attack scenario will be used in Section 6. For simplification we will study only the known plaintext version. It is easy to see that the chosen-plaintext version amounts to the same attack with the number of rounds N_r decreased by approximatively 1 or 2.

Second (specific) attack using the key schedule Another attack we are going to use will require only one known plaintext. However if the key is longer than the block size, we may require another plaintext. This attack is less general and will rely on the fact that the key schedule in Rijndael and Serpent is very similar to the cipher itself: it uses a combination of affine transformations and (the same) S-boxes.

Stronger attack scenarios If such attacks as MQ are possible, i.e. there are efficient methods to solve quadratic equations, then they allow to attack block ciphers in very strong scenarios. For example it is possible to design ciphertext-only attacks. For this we only need to characterize the redundancy of the plaintext in terms of quadratic equations, and this can be done either with partial knowledge of ciphertexts, or with related ciphertexts.

[2] This formula is exact if $Nk = Nb$

4.2 The Direct MQ Attack on Rijndael and Serpent

For example in the second scenario, the problem of recovering the key of the 128-bit Rijndael, will be written as a system of 8000 quadratic equations with 1600 variables. These equations are written in details in Appendix B. In the remaining part of the paper we will study solving such systems of equations. The results for Rijndael are given in Sections 5.2 and 8.1.
Similarly, the 128-bit Serpent would give a system of $(N_r + 1) * B * r + N_r * B * r = 43680$ equations with $(N_r + 1) * s * B + (N_r - 1) * s * B = 8192$ variables.

5 Generic Methods for Solving Multivariate Quadratic Equations

MQ is a known and rather natural NP-hard problem. Several public key cryptosystems are based on MQ, for example HFE [18]. Still, little is known about the actual hardness of it. From the reduction above it is clear that if this problem was very easy for 1600 variables, then Rijndael would be broken. With current attacks, factoring a 1600-bit RSA modulus provides a security level slightly lower than 2^{128} [23]. Therefore if Rijndael is secure, MQ should be at least as hard as factoring.

5.1 Solving MQ with the XL Algorithm

At Crypto'99, Shamir and Kipnis make an important discovery about the MQ problem [21]: Solving it should be much easier for overdefined systems [3]. This idea has been developed and consolidated in a paper published at Eurocrypt'00 [22]. An algorithm called XL is developed for this problem. It seems that for a random system of quadratic equations over $GF(2)$ (or one that looks random) that has a unique solution, the XL method should always work (but maybe not for some very special systems). In [13] T.T. Moh states that "From the theory of Hilbert-Serre, we may deduce that the XL program will work for many interesting cases for D large enough". From [22] it seems also that XL could be subexponential, however very little is known about the actual behaviour of such algorithms for very big systems of equations. Therefore all the complexity estimations we are going to derive in this paper should be considered as approximative. In the Appendix D.2 we recall the XL algorithm and all the basic facts about it from [22].

5.2 First Attempt to Cryptanalyse Rijndael with XL

For the 128-bit Rijndael with 128-bit key, following Section 4.2 (or the Theorem B.3.1 in Appendix B.3), we get a system of $m = 8000$ equations with $n = 1600$ variables. Following the complexity evaluation of XL from [22], (explained also in Appendix D.2), it would lead to a working XL algorithm with the parameter D being about $D \approx n/\sqrt{m} \approx 18$. Thus the complexity of the direct XL attack is about $\binom{n}{D}^{\omega} \approx 2^{330}$.
This attack fails because for a random system of quadratic $R = 8000$ equations with $n = 1600$ variables, we have about $T = n^2/2 \approx 2^{20}$ terms. This gives $R/T \approx 2^{-7.3}$ that is very small and the XL algorithm has to do extensive work in order to achieve $R'/T' \approx 1$. It is easy to see that in our system $T \approx (8 * 32 + 8 * 32 + 8 + 32 + 8) * (N_r * 4 * Nb)$ and this gives only $R/T \approx 2^{-3.5}$, see Appendix B.6. Therefore there **must** be a much better attack.
In the next Section 6.2 we will write such a system of quadratic equations in a different way in order to achieve an even higher value of R/T.

[3] In this paper we will show that if the MQ is sparse, it is still much easier.

6 The (First) XSL Attack

Instead of the general technique XL from [22], we will now design a custom-made algorithm that will take advantage of the specific structure of the equations and of their sparsity. We will call this attack XSL attack which stands for: "e**X**tended **S**parse **L**inearization" or "multiply(**X**) by **S**elected monomials and **L**inearize".
Starting from the initial equations for each S-box of the cipher with r equations and t terms, we will write a set of quadratic equations that will completely define the secret key of the cipher. In the XL algorithm, we would multiply each of these equations by all possible monomials of some degree $D-2$, see Section D.2 or [22]. Instead we will only multiply them by carefully selected monomials. It seems that the best thing to do is to use products of monomials that already appear in other equations. In [22], when $R \geq T$, we have as many equations as the number of terms that appear in these equations and the big system is expected to be solved by adding a new variable for each term, and solving a linear system (doing this is known as linearization).

6.1 The Working condition of the XSL attack or the "T' Method"

There is no need to have R much bigger than T. In the original paper about XL [22], the system was solved when $T - Free$ was a small number. Still it is easy to see that both XL and XSL algorithms work also when $T - Free$ is very big (!). To see this, let for example let x_1 be a variable, and let T' be the number of terms that can be multiplied by x_1 and still belong to the set of T terms. Now we assume that $Free \geq T - T' + C$ with a small C. We apply the following algorithm called "T' method", see Appendix E to see how this works on an explicit example.

1. By one single gaussian elimination we bring the system to a form in which each term is a known linear combination of the terms in T'.
2. We do the same pre-computation two times, for example with T' defined for x_1 and separately for x_2.
3. In each of the two systems, we have a subsystem of C equations that contain only terms of T'. These new equations are probably **not** of the same kind that the initial equations generated in XL-like attacks: only combining all the equations one can obtain some information about the solution, parts of the system usually have many solutions.
4. In each of the two subsystems of exceeding C equations, we multiply each equation by x_1 and respectively x_2. Then we substitute the expressions from point 1 in these to get some **other** equations that contain only terms of T', but for the other variable. These equations are expected to be new and different[4]. First because the equations from point 2 are believed to contain "some information" about the solution that is not in any small subset of R equations, and moreover if we are over $GF(2)$ we will interact with the equation of the field GF(2) that is not necessarily done elsewhere.
5. Thus, if at the beginning $Free >= C + T - T'$ we can "grow" the number of equations. At this moment we expect to have up to $2C$ additional equations, less in practice.
6. We expect that the number of new equations grows exponentially[5].
7. If the initial system has a unique solution we expect that by we will end up with $Free = T$.

[4] We have done several computer simulations, and as expected this heuristic works with good probability. New linearly independent equations are obtained in this way. See also Appendix E for an explicit example.

[5] Even if it grows by 1 each time, the attack will work as predicted.

8. For each equation containing only terms in T', the cost to compute a derived additional equation will be about T'^2. Since there are T' equations missing, we expect to do about T'^3 additional operations in the attack, which can probably be reduced to T'^{ω} and thus will be smaller than T^{ω}.
9. If the whole attack fails one should try with another couple of variables instead of x_1 and x_2, or use three variables from the start (and three systems). We conjecture that three variables should always be sufficient. The number of possibilities grows very fast with the number of variables, a new equation obtained with one variable can be immediately transformed and expanded with **all** the other variables.

For example, in our attack on Rijndael 128 bits given in Section 8.1, we will obtain $T \approx 2^{96}$ and $T' \approx 2^{90}$. The XSL attack is expected to work as long as $Free > T - T' \approx 99.4\%\ T$.

6.2 The Core of the First XSL Attack

Let A be an S-box of a XSL-cipher, called "active S-box". For this S-box A we may write r equations of the form:

$$0 = \sum \alpha_{ijk} X_{i\ j} Y_{i\ k} + \sum \beta_{ij} X_{i\ j} + \sum \gamma_{ij} Y_{i\ j} + \delta.$$

The number of monomials that appear in these equations is small, only t (most of them of the form $X_{i\ j} Y_{i\ k}$). For this reason (unlike as in Appendix B) we kept both the variables $X_{i\ j}$ and $Y_{i\ k}$.
We are going to multiply these equations by one of t monomials existing for some other S-boxes (called "passive" S-boxes). Let S be the total number of S-boxes in our attack. Since we are going to use the most general attack scenario described in 4.1 that ignores the key schedule of the cipher, we consider $N_r + 1$ executions of the cipher and S will be equal to $B * N_r * (N_r + 1)$.
The critical parameter of our attack will be $P \in \mathbb{N}$. In the attack we will multiply each equation of each "active" S-box by all possible terms for all subsets of $(P-1)$ other "passive" S-boxes. The XSL attack is designed in such a way that, for a big P we will obtain something very similar to the general XL attack. However due to the special structure of the equations, a much smaller P should be sufficient.
The total number of equations generated by this method will be about:

$$R \approx r * S * t^{P-1} * \binom{S-1}{P-1}$$

The total number of terms in these equations will be about:

$$T \approx t^P * \binom{S}{P}$$

6.3 Eliminating Obvious Linear Dependencies

It is possible to see that all the set of equations we wrote in Section 6.2 above are not linearly independent. First let us assume $P = 2$. Let $Eq_1 \ldots Eq_r$ and $Eq'_1 \ldots Eq'_r$ be the equations that exist respectively for two S-boxes A and A'. Let $T_1 \ldots T_t$ be the terms that appear in the Eq_i. Instead of writing products: $T_1 Eq'_1, \ldots, T_t Eq'_1$ we may equivalently write the following: $T_1 Eq'_1, \ldots, T_{t-r} Eq'_1$ and then complete by $Eq_1 Eq'_1, \ldots, Eq_r Eq'_1$. But if we apply this transformation for all the equations we have written in the previous section, we see that the each of the $Eq_i Eq'_j$ occurs twice. From this example we see that for any P, one should rather generate the equations of Section 6.2 in the following way: On one hand we restrict to multiplying an "active" equation only by one of the monomials $T_1..T_{t-r}$ for some "passive"

S-box of our system, and on the other hand we also add the equations containing products of several "active" S-boxes. Then it seems that there are no other obvious linear dependencies. The number of equations in the first part of XSL is therefore less than expected:

$$R \approx \sum_{i=1..P} \binom{S}{i} r^i * \binom{S-i}{P-i} (t-r)^{P-i} = \binom{S}{P} \left(t^P - (t-r)^P\right)$$

As before, the total number of terms in these equations is about $T \approx t^P * \binom{S}{P}$.

Remark on R/T

From this we see already that when P grows we will have $R/T \to 1$. Moreover, we have

$$T' \approx t' t^{P-1} * \binom{S-1}{P-1}$$

with $t' < t$ being the number of terms that can be multiplied by x_1, for example $t' = 25$ for Rijndael. In order to solve such a system of equations, following Section 6.1, we need to have $T - R < T'$, i.e.

$$\binom{S}{P} (t-r)^P = \frac{S}{P} \binom{S-1}{P-1} (t-r)^P < t' t^{P-1} \binom{S-1}{P-1}$$

It boils down to $\frac{S}{P} (t-r)^P < t' t^{P-1}$ and already from this we may see that we will have $T - R < T'$ for a sufficiently large P. Moreover, R is not all the equations we will use.

6.4 The Equations on the Diffusion Layers

We do not yet have a system having one and unique solution and we need some additional equations. We will construct these equations in such a way that they can be multiplied by many terms, and still they will be written with the same T monomials.
We will eliminate all the key variables and write additional equations of the form:

$$X_{i\ j} \oplus \sum \alpha_j Y_{i-1\ j} = X'_{i\ j} \oplus \sum \alpha_j Y'_{i-1\ j} = X''_{i\ j} \oplus \sum \alpha_j Y''_{i-1\ j} = \ldots$$

We have $N_r * (N_r+1) * (sB)$ such equations. Each of these equations, called "active equation", will be multiplied by products of terms for some $(P-1)$ "passive" S-boxes. Here we need to exclude the terms for a few neighbouring S-boxes (i.e. that have common variables with the active equation), though some of such terms still can be included and will not add any new terms to the T previously described. The number of new equations is about:

$$R' \approx N_r * (N_r+1) * (sB) * t^{P-1} * \binom{S}{P-1} = S * s * t^{P-1} * \binom{S}{P-1}$$

Again, as in Section 6.3, it is possible to see that one should generate only a part of these equations, the remaining have to be linearly dependent. Thus we will put rather:

$$R' \approx S * s * (t-r)^{P-1} * \binom{S}{P-1}$$

6.5 The Expected Complexity of the XSL Attack(s)

The goal of the attack is to obtain $T - R - R' > T'$. This gives

$$\frac{S}{P} \binom{S-1}{P-1} (t-r)^P - S * s * (t-r)^{P-1} * \binom{S}{P-1} < t' t^{P-1} \binom{S-1}{P-1}$$

$$\frac{S}{P} (t-r)^P < \frac{S^2}{S-P+1} * s * (t-r)^{P-1} + t' t^{P-1}$$

We will assume that $P \ll S$ (S is usually quite big $S \approx BN_r^2$) and thus $S - P + 1 \approx S$.

$$\frac{S}{P}\left(1-\frac{r}{t}\right)^P < S\frac{s}{t}+\frac{t'}{t}$$
$$\left(1-\frac{r}{t}\right)^P < \frac{Ps}{t}+\frac{Pt'}{St}$$
We see that this condition can always be satisfied, and with P that is not too big: the left side decreases exponentially with P, the right side increases. If we consider that $\left(1-\frac{r}{t}\right)^{\frac{t}{r}} \approx 1/e$ we get the following approximation:
$$e^{-P\frac{r}{t}} < \frac{Ps}{t}+\frac{Pt'}{St}$$
$$P > \frac{t}{r}\quad\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right)\qquad (\#)$$
When $r=0$ we will say that $P=\infty$ in the XSL attack: it cannot work then.
If T^ω is the complexity of the Gaussian reduction (see F for details) then the complexity of the XSL attack is about:
$$WF \;=\; T^\omega \;\approx\; t^{\omega P}\binom{S}{P}^{\omega} \;\approx\; (tS)^{\omega P} \;\approx\; \left(t\cdot B\cdot N_r^2\right)^{\omega P} \;\approx\; \left(t/s\cdot Bs\cdot N_r^2\right)^{\omega P} \;\approx$$
$$\approx\; (t/s)^{\omega P}\cdot(B\cdot s\cdot N_r^2)^{\omega P} \;\approx\; (t/s)^{\omega P}\cdot(\text{Block size})^{\omega P}\cdot(\text{Number of rounds})^{2\omega P}$$
Now let us apply the estimation (#). It is easy that the value $\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right)$ is bounded by a constant that does not depend on block size and number of rounds of the cipher. Moreover in practice (for example in our later attacks) we will have the value $\left(-ln\left(\frac{Ps}{t}+\frac{Pt'}{St}\right)\right)$ close to 1. Therefore it is interesting to evaluate the expected complexity of the XSL attack when $P=\lceil t/r\rceil$. It gives the following estimation of the complexity of the XSL attack on block ciphers.
$$WF \;\approx\; (t/s)^{\omega\lceil\frac{t}{r}\rceil+o(1)}\cdot(B\cdot s\cdot N_r^2)^{\omega\lceil\frac{t}{r}\rceil+o(1)} \;\approx\; \Gamma^\omega\cdot\left((\text{Block size})\cdot(\text{Number of rounds})^2\right)^{\omega\lceil\frac{t}{r}\rceil}$$
$$WF=\Gamma^\omega\cdot(\text{Block size})^{\mathcal{O}(\frac{t}{r})}\,(\text{Number of rounds})^{\mathcal{O}(\frac{t}{r})}$$
This is polynomial in the block size and the number of rounds. The constant part depends on Γ that depends only on the parameters of the S-box used in the cipher, and is in general double-exponential in s, see Section 3.1. For a given cipher the constant part Γ^ω in the complexity of XSL will be fixed (but usually very big).

6.6 The Actual Complexity of the XSL Attacks

From the simulations that have been done for XL in [22] and for XSL in Appendix C we believe that XL and XSL attacks will always work for some D (respectively P) and we expect that the XSL attack should give much better results than XL.
In the above derivation we assumed that all the equations in $R+R'$ are linearly independent and this implies that for some fixed P the attack will always work for any number of rounds. From our simulations described in Appendix C it seems that P will rather increase (but slowly) with the number of rounds.
If P were constant, for a fixed S-box that have many overdefined equations, the XSL attack will be polynomial in the number of rounds. Even if P grows slowly, and XSL is subexponential, it would be already **an important breakthrough**, as the classical attacks on block ciphers such as linear or differential cryptanalysis grow exponentially in the number of rounds (and so does the number of required plaintexts).

In fact it is easy to come to conclusion that the problem to break Rijndael is probably subexponential when the number of rounds grows. Indeed, in this paper we show how to write Rijndael as an overdefined system of quadratic equations, with size that is linear in N_r, see Appendix B. The problem of solving such a system of quadratic equations over GF(2) is already believed subexponential (but impractical to solve) with the simple algorithm XL from [22]. See Section B.5 for more comments on this. Finally, our equations from Appendix B are also overdefined and sparse, and this makes thing worse.

7 The Second XSL Attack

The second attack uses the key schedule. Unlike the very general first XSL attack that we studied asymptotically, the second attack is designed to obtain concrete attacks on Rijndael and Serpent.
Let Λ be the number of plaintexts needed in order to completely determine the key used in the cipher. For Rijndael and Serpent we have $\Lambda = 1$ or 2. As before, we will write a system of equations in which a separate variable exists for each input and output bit, of each of the S-boxes, but here **it will also include the S-boxes that are in the key schedule**. We will have:

$$S = \Lambda * B * N_r + D + E,$$

with D being the number of S-boxes in the key schedule and with $E = 0$ or 1 being the number of additional "artificial" S-boxes explained later.
First we will write the equations exactly as described in Sections 6.2 and 6.3. The number of equations in the first part of the attack is again equal to:

$$R \approx \binom{S}{P} \left(t^P - (t-r)^P\right)$$

However here the values of S and the definition of the S-boxes that enter in S has changed, for example the key variables can now be included in t for some of the S-boxes (!). We also have the same formula for T: $T \approx t^P * \binom{S}{P}$.

7.1 The Equations on the Diffusion Layers

The number of key variables used in this attack will be called S_k. We require that:

- The key variables must contain each input bit and each output bit of each of D S-boxes in the key schedule. This gives $S_k = 2 * s * D$ with $D = (L_k - H_k)/s$ for Rijndael and $D = (N_r + 1) * B$ for Serpent.
  - If this is sufficient to linearly span all the key variables, we have $S_k = 2 * s * D$. In this case $E = 0$, i.e. there are no "artificial" S-boxes. This is the case in Serpent.
  - Otherwise, let $E = 1$ and let e be the number of the $K_{i\ j}$ that need to be added to the above $2 * s * D$ variables, in order to linearly span all the the key variables. By inspection we verify that in Rijndael we have $e = 8 * s + 8 * s * 1_{Nk \neq 4}$.
    Here $E = 1$ and we construct an "artificial S-box" in the following way: its equations will be an empty set, i.e. $r = 0$ for this S-box, and its terms will be all the e additional variables. Having one S-box that has a bit different parameters will not change a lot the complexity of our attacks. For example such an artificial S-box is used in our simulations in Appendix C.

Thus for Serpent we have $S_k = 2 * s * D$ and for Rijndael $S_k = 2 * s * D + 8 * s + 8 * s * 1_{Nk \neq 4}$.
We will (as before) denote by $[K_{i\ j}]$ the expression of $K_{i\ j}$ as a linear combination of the S_k "true" key variables. We add the following equations:

$$X_{i+1\ j} = \sum \alpha_j Y_{i\ j} \oplus [K_{i\ j}] \quad \text{for all } i = 0..N_r. \tag{1}$$

Again each of these equations will be multiplied by products of terms of $(P-1)$ "passive" S-boxes (as before chosen out of S without a few "neighbouring"). We obtain a set of equations that use only the T previously described terms[6]. The number of new equations is about: [7]

$$R' \approx \Lambda * s * B * (N_r + 1) * (t-r)^{P-1} * \binom{S}{P-1}$$

7.2 Additional Equations on the Key Schedule

In order to complete the description of the cipher by the equations, and thus get a system having a unique solution we need some more equations. What is missing are the linear equations on the key schedule that come from the fact that our S_k key variables are not all linearly independent. These equations are again multiplied by products of terms of $(P-1)$ "passive" S-boxes. In the case of Rijndael it gives about (again we replaced t by $t-r$):

$$R'' \approx (S_k - L_k) * (t-r)^{P-1} * \binom{S}{P-1}$$

For Serpent we have:

$$R'' \approx (s * D - H_k) * (t-r)^{P-1} * \binom{S}{P-1}$$

7.3 The Complexity of the Second XSL Attack

The attack will work when P is (at least) such that:

$$\frac{R + R' + R''}{T - T'} > 1 \quad (*).$$

For this P, the complexity of the attack is equal to (see also Appendix F): $T^\omega = t^{P\omega} * \binom{S}{P}^\omega$. We will not compute the asymptotic complexity of this attack: it is expected to be very similar to the first XSL attack. Instead we will apply it to concrete ciphers, compute the smallest P value for which the above inequality $(*)$ becomes true, assume that the attack works for this P, and compute the concrete complexity of the attack.

8 The Consequences of the XSL Attacks

8.1 Application to Rijndael

For the basic 128-bit Rijndael, we applied the second XSL attack and only for $P = 8$ we were able to get $\frac{R+R'+R''}{T-T'} = 1.005$. The resulting complexity is much more than the exhaustive search:

$$T^\omega \approx 2^{230}$$

From Section 6.5 it seems that P will not depend on the block and key sizes of the cipher (only the parameters of the S-boxes used). Thus, even if XSL does not break the Rijndael 128 bits, the complexity should not be much higher and break the version with 256-bit key. The detailed computation shows that for $\Lambda = 2$ and $P = 8$ we obtain $\frac{R+R'+R''}{T-T'} = 1.006$ and the complexity evaluation gives:

$$T^\omega \approx 2^{255}$$

---

[6] Unlike the first XSL attack, here the set of S S-boxes have been constructed in such a way that all the $K_{i\ j}$ belong to the set of terms of some S-box.

[7] As in Section 6.4 (and following the ideas from Section 6.3) we have replaced t by $t-r$ in order to avoid to generate too many equations that cannot possibly be linearly independent.

More interesting results can be obtained with cubic equations. Our simulations show that with cubic equations and the Rijndael S-box we have $t = 697$, $r = 471$ and $t' = 242$. Then for $\Lambda = 2$ and $P = 5$ we obtain $\frac{R+R'+R''}{T-T'} = 1.0005$ and the complexity is about:

$$T^{\omega} \approx 2^{203}$$

Even if we assume that the Gaussian reduction is cubic, we still get 2^{250}, which is less than the exhaustive search. We obtain also that for $P = 6$ and $P = 7$ the complexity is respectively 2^{240} and 2^{278}.

8.2 Application to Serpent

For Serpent we obtain exactly the same results for the key length 128, 192 and 256 bits (the XSL attacks works by thresholds). Thus for $P = 4$ we get $\frac{R+R'+R''}{T-T'}$ equal respectively to 1.0007, 1.0004 and 1.0001. The complexity of the attack is about:

$$T^{\omega} \approx 2^{143}$$

It seems that the XSL attack will break Serpent for key lengths 192 and 256 bits. Moreover, this will hold also if the Gaussian reduction is cubic and gives still only 2^{175}. We obtain also that for $P = 5, 6, 7, 8$ the complexity is respectively $2^{176}, 2^{208}, 2^{240}$ and 2^{272}.

8.3 How Realistic is the XSL Attack ?

Though XSL attacks certainly will work for some P, we considered the minimum value P for which $\frac{R+R'+R''}{T-T'} \geq 1$. A small change (e.g. increase by 1 or 2) in P leads to an important overload in the complexity. The condition $\frac{R+R'+R''}{T-T'} \geq 1$ is necessary, but not sufficient. In order to test the actual behaviour of the XSL attacks, in Appendix C we give the description and results we obtained running the XSL attack on a "toy cipher". These simulations show that P will probably increase, but very slowly, with the number of rounds.

8.4 Consequences for the Design of Block Ciphers

There are two complementary approaches in the block cipher design that could be seen in the AES contest. Either a cipher is designed with a very small number of rounds that are very complex (for example in DFC), or it has a large number of rounds that are very simple (for example in Serpent).

In [26] the authors warn that: "an attack against Serpent may hold for any set of (random) S-boxes". It seems that we have found such an attack. We claim therefore that using many layers of very simple S-boxes is not a very good idea, and is susceptible to attacks with a complexity growing slowly in the number of rounds (with a huge constant). Still, a correct choice of parameters will prevent the attacks.

For different reasons, the XSL attack is also applicable to all ciphers in which the only non-linear part is the inverse function in $GF(2^s)$, with a small s. Therefore ciphers such as Rijndael and Camellia should either use s that is sufficiently large, for example $s = 16$, or consider different S-boxes. This last possibility should give new optimal designs of S-boxes, not only close to optimal in terms of linear and differential attacks, but also incorporating our new criterion, i.e. having a big value of Γ, for example $\Gamma > 2^{20}$.

Even if the attacks of the present paper have not yet been tested on really big examples, they are an important threat for ciphers such as Rijndael, Serpent and Camellia. We propose that all block ciphers should apply the following criterion (due originally to Shannon [24]):

The attacker should not be able to write a system of algebraic equations of simple type and of any reasonable size, that completely characterizes the secret key.

An immediate way to achieve this is to use at least a few (relatively) big randomly generated S-boxes. In the future the XSL attack should be taken into account in the design of new kinds of S-boxes.

9 Conclusion

In this paper we point out an unexpected property of Rijndael and Serpent: they can be described as a system of overdefined and sparse quadratic equations over $GF(2)$. It was known from Eurocrypt'00 that solving such systems is easier if they are overdefined, and the problem has been conjectured to be subexponential for small fields such as $GF(2)$. From this argument we obtain that the security of Rijndael and Serpent probably **does not grow exponentially** with the number of rounds.

A direct application of the XL attack from Eurocrypt'00 is extremely inefficient. Knowing that the equations are not only overdefined, but also sparse and structured, we have introduced a new method called XSL. If the XSL attack works as well predicted, it seems that it could even be polynomial in the number of rounds of the cipher. It seems also to break Rijndael 256 bits and Serpent for key lengths 192 and 256 bits. In order to prevent such attacks, we propose that at least a few S-boxes in a cipher should not be described by a small system of overdefined multivariate equations.

References

1. Ross Anderson, Eli Biham and Lars Knudsen: *Serpent: A Proposal for the Advanced Encryption Standard.* Available from `http://www.cl.cam.ac.uk/~rja14/serpent.html`
2. Anne Canteaut, Marion Videau: *Degree of composition of highly nonlinear functions and applications to higher order differential cryptanalysis;* Eurocrypt 2002, LNCS 2332, Springer.
3. Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions"; J. Symbolic Computation (1990), 9, pp. 251-280.
4. Joan Daemen, Vincent Rijmen: *AES proposal: Rijndael;* The latest revised version of the proposal is available on the internet, `http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf`
5. Nicolas Courtois, Louis Goubin, Willi Meier, Jean-Daniel Tacier: *Solving Underdefined Systems of Multivariate Quadratic Equations;* PKC 2002, LNCS 2274, Springer, pp. 211-227.
6. Nicolas Courtois: *The security of Hidden Field Equations (HFE)*; Cryptographers' Track Rsa Conference 2001, San Francisco 8-12 April 2001, LNCS2020, Springer-Verlag, pp. 266-281.
7. Horst Feistel: *Cryptography and computer privacy;* Scientific American, vol. 228, No. 5, pp. 15-23, May 1973.
8. Niels Ferguson, Richard Schroeppel and Doug Whiting: *A simple algebraic representation of Rijndael;* Draft 2001/05/16, presented at the rump session of Crypto 2000 and available at `http://www.macfergus.com/niels/pubs/rdalgeq.html`.
9. Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, Doug Whiting: Improved Cryptanalysis of Rijndael, FSE 2000, Springer.
10. J.B. Kam and G.I. Davida: *Structured design of substitution-permutation encryption networks;* IEEE Trans. on Computers, Vol. C-28, 1979, pp.747-753.
11. Lars R. Knudsen, Vincent Rijmen: *On the Decorrelated Fast Cipher (DFC) and its Theory*; FSE'99, Springer, LNCS 1636, pp. 81-94.
12. Michael Luby, Charles W. Rackoff, *How to construct pseudorandom permutations from pseudorandom functions;* , SIAM Journal on Computing, vol. 17, n. 2, pp. 373-386, April 1988.
13. T.T. Moh: *On The Method of XL and Its Inefficiency Against TTM*, available at `http://eprint.iacr.org/2001/047/`.
14. Moni Naor and Omer Reingold: *On the construction of pseudo-random permutations: Luby-Rackoff revisited;* Journal of Cryptology, vol 12, 1999, pp. 29-66.
15. Kaisa Nyberg: *Differentially Uniform Mappings for Cryptography;* Eurocrypt'93, LNCS 765, Springer, pp. 55-64.
16. Jacques Patarin: *Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88*; Crypto'95, Springer-Verlag, pp. 248-261.
17. Jacques Patarin: *Generic Attacks on Feistel Schemes* ; Asiacrypt 2001, LNCS 2248, Springer, pp. 222-238.
18. Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms;* in Eurocrypt'96, Springer Verlag, pp. 33-48.
19. Jacques Patarin, Nicolas Courtois, Louis Goubin: *Improved Algorithms for Isomorphism of Polynomials*; Eurocrypt 1998, Springer-Verlag.
20. Adi Shamir, Alex Biryukov: *Structural Cryptanalysis of SASAS;* Eurocrypt 2001, LNCS 2045, Springer, pp. 394-405.
21. Adi Shamir, Aviad Kipnis: *Cryptanalysis of the HFE Public Key Cryptosystem*; In Advances in Cryptology, Proceedings of Crypto'99, Springer-Verlag, LNCS.
22. Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407.
23. Robert D. Silverman: *A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths;* RSA Lab. report, `http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html`.
24. Claude Elwood Shannon: *Communication theory of secrecy systems;* , Bell System Technical Journal 28 (1949), see in patricular page 704.
25. Serge Vaudenay: *Provable Security for Block Ciphers by Decorrelation;* Technical Report LIENS-98-8 of the Laboratoire d'Informatique de l'Ecole Normale Supérieure, 1998. Available at `http://lasecwww.epfl.ch/query.msql?ref=Vau98b`.
26. Serge Vaudenay, Shiho Moriai: *On the Pseudorandomness of Top-Level Schemes of Block Ciphers;* Asiacrypt 2000, LNCS 1976, Springer, pp. 289-302.

A More on Algebraic Properties of the Rijndael S-box

Rijndael handles most of its computations in $GF(256)$ that is represented, on one hand by polynomials $b_7X^7 + \ldots + b_1X + b_0$ in $GF(2)[X]/X^8 + X^4 + X^3 + X + 1$, and on the other hand by bytes written in hexadecimal notation corresponding to the number $b_7 2^7 + \ldots + b_1 2^1 + b_0$. For example "03" is the polynomial X+1 in $GF(2)[X]/X^8 + X^4 + X^3 + X + 1$.
Rijndael S-box is a composition of the "patched" inverse in GF(256) with 0 mapped on itself, with a multivariate affine transformation $GF(2)^8 \to GF(2)^8$. Following [4] we call these functions respectively g and f and we call $S = f \circ g$.
We note x an input value and $y = g(x)$ the corresponding output value. We will also note $z = S(x) = f(g(x)) = f(y)$.
A more elegant way of representing g is to write it as the power function.. It is easy to see that we have $g : x \mapsto x^{254} \bmod X^8 + X^4 + X^3 + X + 1$, as $254 \equiv -1 \bmod \left(2^8 - 1\right)$. In this representation we don't need to handle a special case of 0. The multivariate affine function $f : GF(2)^8 \to GF(2)^8$ can also be written as a linearized polynomial $f : GF(2^8) \to GF(2^8)$:

$$z = f(y) = \text{"63"} + \text{"05"}y + \text{"09"}y^2 + \text{"}f9\text{"}y^4 + \text{"25"}y^8 + \text{"}f4\text{"}y^{16} + \text{"01"}y^{32} + \text{"}b5\text{"}y^{64} + \text{"}8f\text{"}y^{128}$$

The composition $S = f \circ g$ gives the following sparse polynomial:

$$z = S(x) = f(g(x)) = f(y) = f(x^{254})$$

$$z = S(x) =$$

$$\text{"63"} + \text{"}8f\text{"}x^{127} + \text{"}b5\text{"}x^{191} + \text{"01"}x^{123} + \text{"}f4\text{"}x^{239} + \text{"25"}x^{247} + \text{"}f9\text{"}x^{251} + \text{"09"}x^{253} + \text{"05"}x^{254}$$

From the definition of S, we have:

$$\forall x \neq 0 \qquad 1 = xy$$

This equation gives in turn 8 bi-linear equations in 8 variables. We will not write these equations between the x_i and the y_j, but instead we will write directly the resulting equations between the inputs and outputs of the whole S-box:

$$\begin{cases}
0 = z_0x_4 + z_0x_5 + z_0x_1 + x_0z_6 + x_0z_4 + x_0z_1 + x_2z_7 + x_2z_4 + x_2z_2 + x_3z_6 + x_3z_3 + x_3z_1 + \\
\quad x_4z_6 + x_4z_5 + x_4z_4 + x_4z_2 + x_4z_1 + x_5z_6 + x_5z_7 + x_5z_5 + x_5z_3 + x_6z_6 + x_6z_7 + x_6z_5 + \\
\quad x_6z_4 + x_6z_2 + x_7z_5 + x_7z_3 + x_1z_5 + x_1z_3 + x_5 + x_7 \\
0 = z_0x_0 + z_0x_3 + z_0x_4 + x_0z_5 + x_0z_3 + x_2z_6 + x_2z_3 + x_2z_1 + x_3z_6 + x_3z_5 + x_3z_4 + x_3z_2 + \\
\quad x_3z_1 + x_4z_6 + x_4z_7 + x_4z_5 + x_4z_3 + x_5z_6 + x_5z_7 + x_5z_5 + x_5z_4 + x_5z_2 + x_6z_5 + x_6z_3 + \\
\quad x_7z_6 + x_7z_2 + x_7z_1 + x_1z_7 + x_1z_4 + x_1z_2 + x_4 + x_6 \\
0 = z_0x_2 + z_0x_3 + z_0x_7 + x_0z_7 + x_0z_4 + x_0z_2 + x_2z_6 + x_2z_5 + x_2z_4 + x_2z_2 + x_2z_1 + x_3z_6 + \\
\quad x_3z_7 + x_3z_5 + x_3z_3 + x_4z_6 + x_4z_7 + x_4z_5 + x_4z_4 + x_4z_2 + x_5z_5 + x_5z_3 + x_6z_6 + x_6z_2 + \\
\quad x_6z_1 + x_7z_5 + x_7z_1 + x_1z_6 + x_1z_3 + x_1z_1 + x_3 + x_5 + x_7 \\
0 = z_0x_2 + z_0x_6 + z_0x_7 + z_0x_1 + x_0z_6 + x_0z_3 + x_0z_1 + x_2z_6 + x_2z_7 + x_2z_5 + x_2z_3 + x_3z_6 + \\
\quad x_3z_7 + x_3z_5 + x_3z_4 + x_3z_2 + x_4z_5 + x_4z_3 + x_5z_6 + x_5z_2 + x_5z_1 + x_6z_5 + x_6z_1 + x_7z_6 + \\
\quad x_7z_7 + x_7z_1 + x_1z_6 + x_1z_5 + x_1z_4 + x_1z_2 + x_1z_1 + x_2 + x_4 + x_6 + x_7 \\
0 = z_0x_0 + z_0x_4 + z_0x_6 + z_0x_7 + x_0z_5 + x_0z_2 + x_2z_6 + x_2z_5 + x_3z_6 + x_3z_5 + x_3z_1 + x_4z_5 + \\
\quad x_4z_4 + x_5z_6 + x_5z_7 + x_5z_3 + x_5z_1 + x_6z_5 + x_6z_4 + x_6z_2 + x_6z_1 + x_7z_6 + x_7z_7 + x_7z_3 + \\
\quad x_1z_6 + x_1z_7 + x_3 + x_6 + x_1 \\
0 = z_0x_3 + z_0x_4 + z_0x_6 + z_0x_1 + x_0z_7 + x_0z_4 + x_0z_1 + x_2z_6 + x_2z_7 + x_2z_5 + x_2z_4 + x_2z_2 + \\
\quad x_2z_1 + x_3z_6 + x_3z_5 + x_3z_4 + x_3z_3 + x_3z_1 + x_4z_7 + x_4z_5 + x_4z_4 + x_4z_3 + x_4z_2 + x_5z_6 + \\
\quad x_5z_7 + x_5z_4 + x_5z_3 + x_5z_2 + x_5z_1 + x_6z_5 + x_6z_4 + x_6z_3 + x_6z_2 + x_7z_7 + x_7z_4 + x_7z_3 + \\
\quad x_7z_2 + x_7z_1 + x_1z_6 + x_1z_3 + x_0 + x_2 + x_7 \\
0 = z_0x_0 + z_0x_2 + z_0x_3 + z_0x_5 + z_0x_7 + x_0z_6 + x_0z_3 + x_2z_6 + x_2z_5 + x_2z_4 + x_2z_3 + x_2z_1 + \\
\quad x_3z_7 + x_3z_5 + x_3z_4 + x_3z_3 + x_3z_2 + x_4z_6 + x_4z_7 + x_4z_4 + x_4z_3 + x_4z_2 + x_4z_1 + x_5z_5 + \\
\quad x_5z_4 + x_5z_3 + x_5z_2 + x_6z_7 + x_6z_4 + x_6z_3 + x_6z_2 + x_6z_1 + x_7z_4 + x_7z_3 + x_7z_2 + x_1z_6 + \\
\quad x_1z_7 + x_1z_5 + x_1z_4 + x_1z_2 + x_1z_1 + x_6 + x_7 + x_1 \\
1 = x_0 + x_6 + z_0x_2 + z_0x_5 + z_0x_6 + x_0z_7 + x_0z_5 + x_0z_2 + x_2z_5 + x_2z_3 + x_3z_7 + x_3z_4 + x_3z_2 + \\
\quad x_4z_6 + x_4z_3 + x_4z_1 + x_5z_6 + x_5z_5 + x_5z_4 + x_5z_2 + x_5z_1 + x_6z_6 + x_6z_7 + x_6z_5 + x_6z_3 + \\
\quad x_7z_6 + x_7z_7 + x_7z_5 + x_7z_4 + x_7z_2 + x_1z_6 + x_1z_4 + x_1z_1
\end{cases}$$

We observe that the first 7 equations have no constant parts and therefore are also true for $x = 0$. Therefore we obtained here 7 equations that are true with probability 1, plus one additional equation that is true if and only if $x \neq 0$, i.e. with probability $255/256$.
The existence of these (quadratic) equations for g and S is obvious. Surprisingly, we will show that much more such equations exist. (It leads to systems of equations that have much more equations than unknowns, and allows interesting attacks on Rijndael.)
We observe that we have:

$$\forall x \neq 0 \qquad x = x^2 * y$$

This equation happens to be true also for $x = 0$. Wa have therefore:

$$\forall x \in GF(256) \qquad \begin{cases} x = x^2 * y \\ x^2 = x^4 * y^2 \\ \qquad \vdots \\ x^{128} = x * y^{128} \end{cases}$$

Each of equation is the square of the previous one, and since the square is linear as a multivariate function, each these 8 equations will generate **the same** set (more precisely the same modulo a linear combination) of 8 multivariate equations on the x_i and the y_j.
We choose therefore one of these equations, for example the last. It is symmetric with respect to the exchange of x and y and we obtain the following 2 equations:

$$\begin{cases} x^{128} = xy^{128} \\ y^{128} = yx^{128} \end{cases}$$

We have two equations in GF(256) are true with probability 1. Since $x \mapsto x^{128}$ is linear, if written as a set of 8 multivariate linear functions, each of above 2 equations will give 8 quadratic equations with 8 variables. We compute directly the resulting equations on the whole S-box:

$$\begin{cases}
0 = x_3+x_5+x_6+x_1+x_2z_2+x_5z_7+x_7z_4+x_7z_1+x_7z_3+x_0z_1+x_6z_5+x_6z_3+x_7z_7+x_4z_6+ \\
\quad x_4z_1+x_4z_5+x_4z_0+x_4z_2+x_1z_5+x_1z_3+x_5z_5+x_5z_3+x_5z_0+x_3z_1+x_3z_3+x_6z_6+x_3z_4+ \\
\quad x_2z_3+x_2z_6+x_4z_7+x_0z_5+x_0z_3+x_1z_4+x_1z_7+x_6z_1+x_3z_0+x_4z_3+x_0z_7+x_1z_6+x_2z_5 \\
0 = x_3+x_6+x_1+x_2z_4+x_5z_1+x_7z_1+x_5z_6+x_0z_6+x_0z_4+x_6z_3+x_6z_4+x_6z_7+x_7z_7+ \\
\quad x_7z_5+x_7z_2+x_4z_5+x_4z_0+x_1z_5+x_1z_3+x_5z_5+x_5z_3+x_3z_1+x_3z_3+x_3z_6+x_2z_1+x_2z_3+ \\
\quad x_4z_7+x_0z_5+x_0z_3+x_1z_2+x_6z_1+x_3z_5+x_3z_0+x_3z_2+x_4z_3+x_0z_7+x_3z_7+x_1z_6+x_2z_0 \\
0 = x_3+x_4+x_5+x_1+x_2z_2+x_2z_7+x_5z_1+x_5z_4+x_5z_7+x_7z_6+x_7z_4+x_7z_1+x_0z_6+x_6z_5+ \\
\quad x_6z_2+x_6z_7+x_7z_7+x_4z_6+x_4z_1+x_4z_5+x_1z_3+x_1z_0+x_5z_3+x_3z_3+x_2z_1+x_2z_3+ \\
\quad x_2z_6+x_0z_5+x_0z_3+x_1z_4+x_6z_1+x_3z_5+x_3z_0+x_4z_3+x_0z_2+x_3z_7+x_1z_1+x_2z_5+x_2z_0 \\
0 = x_3+x_4+x_1+x_2z_7+x_5z_1+x_5z_7+x_7z_4+x_0z_4+x_0z_1+x_6z_4+x_6z_7+x_7z_7+x_7z_5+ \\
\quad x_7z_2+x_4z_4+x_4z_1+x_1z_5+x_1z_3+x_1z_0+x_5z_5+x_3z_1+x_3z_3+x_3z_6+x_6z_6+x_5z_2+ \\
\quad x_2z_3+x_4z_7+x_0z_3+x_0z_0+x_1z_2+x_1z_7+x_6z_1+x_3z_5+x_4z_3+x_1z_1+x_1z_6+x_2z_5+x_2z_0 \\
0 = x_2+x_6+x_7+x_1+x_2z_2+x_5z_1+x_5z_4+x_7z_4+x_7z_1+x_5z_6+x_7z_3+x_0z_6+x_6z_3+ \\
\quad x_6z_2+x_6z_4+x_6z_7+x_7z_7+x_7z_2+x_4z_6+x_4z_0+x_1z_0+x_5z_5+x_5z_3+x_5z_0+x_6z_6+ \\
\quad x_2z_1+x_0z_0+x_1z_4+x_6z_1+x_3z_0+x_4z_3+x_0z_2+x_3z_7+x_1z_6 \\
0 = x_2+x_3+x_4+x_5+x_1+x_2z_2+x_2z_7+x_5z_1+x_5z_4+x_7z_6+x_7z_1+x_5z_6+x_0z_6+ \\
\quad x_0z_4+x_0z_1+x_6z_5+x_6z_2+x_6z_4+x_6z_7+x_7z_2+x_4z_4+x_4z_2+x_1z_5+x_1z_3+x_5z_5+ \\
\quad x_5z_0+x_3z_1+x_3z_6+x_6z_6+x_5z_2+x_3z_4+x_2z_3+x_2z_6+x_4z_7+x_0z_5+x_0z_3+x_0z_0+ \\
\quad x_1z_2+x_1z_4+x_1z_7+x_0z_7+x_1z_1+x_1z_6+x_2z_5+x_2z_0 \\
0 = x_0+x_2+x_3+x_7+x_2z_4+x_5z_4+x_5z_7+x_7z_6+x_7z_1+x_5z_6+x_0z_6+x_0z_4+x_0z_1+ \\
\quad x_6z_2+x_7z_7+x_4z_6+x_4z_4+x_4z_1+x_4z_5+x_4z_0+x_4z_2+x_1z_5+x_1z_3+x_1z_0+x_5z_5+ \\
\quad x_6z_6+x_5z_2+x_3z_4+x_2z_1+x_2z_6+x_7z_0+x_0z_5+x_0z_3+x_1z_2+x_1z_7+x_6z_1+x_3z_2+ \\
\quad x_0z_2+x_0z_7+x_3z_7+x_1z_6 \\
0 = x_3+x_5+x_2z_4+x_2z_7+x_5z_1+x_5z_7+x_7z_6+x_7z_1+x_5z_6+x_7z_3+x_0z_6+x_0z_1+x_6z_5+ \\
\quad x_6z_3+x_6z_0+x_6z_7+x_7z_5+x_4z_4+x_4z_1+x_4z_0+x_1z_5+x_1z_3+x_5z_5+x_5z_3+x_5z_0+ \\
\quad x_3z_3+x_3z_6+x_5z_2+x_2z_3+x_2z_6+x_0z_0+x_1z_7+x_3z_5+x_3z_2+x_4z_3+x_0z_2+x_1z_1+x_2z_5
\end{cases}$$

$$
\begin{cases}
0 = x_5+x_7+z_7+z_5+z_3+z_1+x_5z_1+x_5z_4+x_7z_3+x_0z_6+x_0z_4+x_0z_1+x_6z_3+x_7z_2+x_4z_4+ \\
\quad x_4z_2+x_1z_5+x_1z_0+x_5z_3+x_6z_6+x_3z_4+x_2z_3+x_4z_7+x_7z_0+x_6z_1+x_3z_7+x_2z_5+x_2z_0 \\
0 = x_3+x_5+x_7+z_6+z_7+z_5+z_4+z_3+x_2z_2+x_2z_4+x_2z_7+x_7z_1+x_6z_5+x_6z_0+ \\
\quad x_6z_2+x_6z_4+x_7z_7+x_7z_2+x_4z_6+x_4z_1+x_5z_3+x_5z_0+x_3z_1+x_3z_3+x_6z_6+x_5z_2+ \\
\quad x_3z_4+x_0z_5+x_0z_3+x_0z_0+x_1z_4+x_1z_7+x_6z_1+x_4z_3 \\
0 = x_3+x_5+x_6+x_7+x_1+z_6+z_5+z_3+z_2+x_5z_1+x_5z_7+x_7z_6+x_7z_1+x_0z_4+x_6z_5+ \\
\quad x_6z_3+x_6z_0+x_6z_7+x_4z_6+x_4z_4+x_4z_1+x_4z_5+x_4z_0+x_4z_2+x_1z_3+x_3z_3+x_6z_6+ \\
\quad x_5z_2+x_2z_1+x_2z_3+x_2z_6+x_7z_0+x_1z_4+x_3z_0+x_3z_2+x_0z_2+x_0z_7+x_1z_1 \\
0 = x_3+x_4+x_5+x_1+z_4+z_3+z_1+z_0+x_2z_2+x_2z_4+x_5z_1+x_5z_6+x_0z_6+x_0z_1+x_6z_5+ \\
\quad x_6z_2+x_6z_4+x_6z_7+x_7z_7+x_7z_5+x_4z_6+x_4z_5+x_4z_0+x_1z_3+x_1z_0+x_5z_0+x_3z_1+ \\
\quad x_6z_6+x_2z_1+x_2z_6+x_4z_7+x_7z_0+x_0z_3+x_1z_2+x_3z_2+x_4z_3+x_3z_7+x_2z_5+x_2z_0 \\
0 = x_2+x_3+x_5+x_6+x_1+z_6+z_2+z_0+x_2z_7+x_5z_1+x_5z_4+x_5z_7+x_7z_6+x_7z_4+x_7z_3+ \\
\quad x_6z_5+x_7z_7+x_7z_2+x_4z_6+x_4z_5+x_1z_5+x_1z_0+x_5z_5+x_5z_3+x_5z_0+x_3z_1+x_3z_6+ \\
\quad x_6z_6+x_3z_4+x_2z_6+x_7z_0+x_0z_5+x_0z_0+x_1z_2+x_1z_7+x_6z_1+x_3z_0+x_0z_2+x_3z_7+x_1z_1 \\
0 = x_0+x_3+x_4+x_5+x_1+z_6+z_7+z_5+z_4+z_3+z_1+z_0+x_5z_1+x_5z_7+x_7z_4+x_5z_6+ \\
\quad x_0z_4+x_0z_1+x_6z_5+x_6z_3+x_6z_0+x_6z_4+x_7z_7+x_7z_5+x_4z_6+x_4z_4+x_4z_1+x_4z_5+ \\
\quad x_4z_2+x_1z_5+x_5z_0+x_3z_1+x_3z_3+x_6z_6+x_5z_2+x_2z_3+x_2z_6+x_4z_7+x_7z_0+x_1z_4+ \\
\quad x_1z_7+x_6z_1+x_3z_5+x_3z_0+x_0z_7+x_1z_1+x_1z_6+x_2z_0 \\
0 = x_2+x_3+x_7+x_1+z_6+z_7+z_5+z_4+z_3+z_2+z_1+1+x_2z_2+x_2z_4+x_2z_7+x_5z_4+ \\
\quad x_5z_7+x_7z_1+x_7z_3+x_0z_6+x_6z_5+x_6z_3+x_6z_0+x_6z_2+x_6z_4+x_6z_7+x_7z_7+x_4z_6+ \\
\quad x_4z_4+x_4z_1+x_4z_5+x_4z_0+x_1z_5+x_1z_3+x_1z_0+x_5z_5+x_5z_0+x_3z_1+x_3z_6+x_2z_1+ \\
\quad x_2z_6+x_0z_3+x_0z_0+x_3z_0+x_3z_2+x_4z_3+x_3z_7+x_1z_1+x_2z_5 \\
0 = x_0+x_7+x_1+z_6+z_2+z_1+z_0+1+x_2z_4+x_5z_4+x_5z_7+x_7z_4+x_7z_1+x_7z_3+x_6z_2+ \\
\quad x_6z_4+x_6z_7+x_4z_5+x_4z_0+x_1z_5+x_2z_1+x_2z_6+x_0z_5+x_1z_2+x_1z_7+x_3z_5+x_3z_0+ \\
\quad x_4z_3+x_0z_2+x_0z_7+x_1z_1+x_1z_6
\end{cases}
$$

In all, for each Rijndael S-box we have 23 bi-affine equations between the x_i and the z_j. We have verified that all these equations are linearly independent and that there are no more such equations.

Moreover, if x is always different than 0, we will have all the 24 linearly independent equations that will be satisfied.

A.1 Remarks

Fully quadratic equations. It is possible to see that if we consider fully quadratic equations, not only bi-affine, for each S-box of Rijndael there are $r = 39$ quadratic equations with $t = 137$. The additional 16 equations come from the following two equations:

$$\begin{cases} x^4y = x^3 \\ y^4x = y^3 \end{cases}$$

However when using $r = 39$ and $t = 137$ we always obtained worse results in the XSL attack than with $r = 23$ and $t = 81$. This is due to the fact that it gives $\Gamma = 2^{16.4}$ instead of $2^{13.4}$.

About the inverse-based S-boxes in general Similarly, it is easy to see that if the S-box on s bits is an affine transformation of the inverse function in $GF(2^s)$, then it will give $3s-1$ bi-affine equations true with probability 1, and one additional equation true with probability $1 - \frac{1}{2^s}$. We conjecture that there is no more such equations.

Up till now, it seemed a very good idea to use such S-boxes in practical ciphers. This was due to the fact that the inverse function (and its affine equivalents) has many **optimality** results with regard to linear, differential and high-order differential attacks, see [2, 15].

We have done computer simulations for many permutations including all the possible powers in $GF(2^s)$. They showed that the inverse (and its equivalents) is the **worse** in terms of the number of such bi-affine equations. It is an open problem to find **any other** non-linear function $GF(2^s) \to GF(2^s)$ that admits so many equations, for some $s > 0$. Therefore though in many cases the ciphers are probably still very secure, we do not advocate to use such S-boxes.

Related work: The equations we have found for the Rijndael S-box are exactly of the same type and of very similar origin, as the equations that Jacques Patarin have discovered in 1988 on the Matsumoto-Imai cryptosystem [16]. The existence of such equations on Rijndael S-boxes have been first discovered (but not published) by Nicolas Courtois, Louis Goubin and Jacques Patarin, as soon as Rijndael have been proposed as AES in 2000.

B The Direct MQ Attack on Rijndael

It is interesting to know how to describe Rijndael as a system of quadratic equations with a minimum number of variables and maximum number of equations. We are in the second attack scenario with one or a few known plaintexts, as in Section 4.1.

B.1 Minimizing the Number of Variables for Rijndael

For each round i, we know that there are $r * 4 * Nb$ quadratic equations between the $(Z_{i-1\ j} + K_{i-1\ j})$ and the $(Z_{i\ k})$. They are of the following form:

$$0 = \sum \alpha_{ijk} Z_{i-1\ j} Z_{i\ k} + \sum \alpha_{ijk} [K_{i-1\ j}] Z_{i\ k} + \sum \beta_{ij} Z_{i\ j} + \sum \beta_{ij} [K_{i\ j}] + \gamma.$$

Exception is made for the first round, for which the Z_0 being known, they are of the form:

$$0 = \sum \alpha_{ij} [K_{0\ i}] Z_{1\ j} + \sum \beta_i Z_{1\ i} + \sum \gamma_i [K_{0\ i}] + \delta.$$

Finally, for the last round, the $X_{N_r\ k}$ will be expressed as a sum of the known ciphertext $Z_{N_r+1\ k}$ and $[K_{N_r\ k}]$, giving the equations of the form:

$$0 = \sum \alpha_{ij} Z_{N_r-1\ i} [K_{N_r\ j}] + \sum \alpha_{ij} [K_{N_r-1\ i}][K_{N_r\ j}] + \sum \beta_i Z_{N_r-1\ i} +$$
$$+ \sum \beta_i [K_{N_r-1\ i}] + \sum \gamma_i [K_{N_r\ i}] + \delta.$$

In all we will get $4 * r * N_r * Nb$ quadratic equations over $GF(2)$. The number of variables $Z_{i\ j}$ is only $4 * s * (N_r - 1) * Nb$.

B.2 Using the Key Schedule

In the cipher we have:

$$X_{i+1\ j} = Z_{i\ j} \oplus [K_{i\ j}] \quad \text{for all } i = 0..N_r. \tag{2}$$

In order to define what are the $[K_{i\ j}]$ we need to choose a basis for the $K_{i\ j}$. From the key schedule [4] it is obvious that one may take as "true key variables" all the Nk variables from the first round, then all the first columns of each consecutive key states, and if $Nk = 8$, also the 5th columns. By inspection we see that the number of "true key variables" is:

$$L_k = \begin{cases} 32 * \left(Nk + \lceil \frac{N_r * Nb + Nb - Nk}{Nk} \rceil\right) & \text{if } Nk \neq 8 \\ 32 * \left(Nk + \lceil \frac{N_r * Nb + Nb - Nk}{4} \rceil\right) & \text{if } Nk = 8 \end{cases}$$

For example, for 128-bit Rijndael with $H_k = 128$ we have $L_k = 32 * (4 + 10) = 448$ "true" key variables.

Additional equations. We call "redundant true variables" all the $L_k - H_k$ additional variables that are determined by some initial subset of H_k variables. From the key schedule we see that for each of these $L_k - H_k$ "redundant true variables" we may write $r = 23$ (or 24) quadratic equations. Each of the "redundant true" key state columns is a XOR of one the previous columns, a parallel application of 4 S-boxes to another column, and of a constant. Thus these equations are of the form:

$$\sum \alpha_{ijkl} [K_{i\ j}][K_{k\ l}] + \sum \beta_{ij} [K_{i\ j}] + \gamma. \tag{3}$$

The number of these equations is:

$$r * \frac{L_k - H_k}{s}$$

B.3 Putting all the Equations Together

Theorem B.3.1 (Reduction Rijndael $\rightarrow$ MQ). The problem of recovering the secret key of Rijndael given about one pair plaintext/ciphertext can be written as an overdefined system of

$$m = 4 * r * Nb * N_r + r(L_k - H_k)/s$$

sparse quadratic equations with the number of unknowns being:

$$n = 4 * s * (N_r - 1) * Nb + L_k.$$

B.4 Examples

We will use fully quadratic equations obtained in Section A.1. We have $r = 39$ and $t = 137$, however since this attack will only require 1 or 2 known plaintexts, we may assume $r = 40$ (exactly as in Section 3.3).
Thus for the 128-bit Rijndael with 128-bit key, we can write the problem of recovering the key as a system of 8000 quadratic equations with 1600 variables.
For the 256-bit Rijndael with 256-bit key, we get a system of 22400 quadratic equations with 4480 variables.

B.5 Theoretical Consequences for Rijndael and AES

The above reduction has already some very important consequences for Rijndael and AES. We consider the security of some generalized version of Rijndael in which the number of rounds N_r increases and all the other parameters are fixed.
On one hand, in all general attacks previously known against such ciphers, for example in linear or differential attacks, the security grows exponentially with N_r. There are also combinatorial attacks such as square attack, but these will simply not work if N_r is sufficiently large.
On the other hand, we observe that the number of variables (and the number of equations) in the reduction is **linear** in the number of rounds N_r. Therefore, if the MQ problem is subexponential, which is our view of the results given in the XL paper [22], to break Rijndael will also be subexponential[8], i.e. the security will **not** grow exponentially with the number of rounds N_r.
Remark: It is important to see that the result would not be the same if the reduction were for example quadratic in Nr. In this case XL could be subexponential, for example in $n^{\sqrt{n}}$ but the Rijndael could still be fully exponential, for example in $(N_r^2)^{N_r}$.
Remark 2: It seems that the same remark will hold for any block cipher composed with rounds of fixed type: obviously each of them can always be written as a set of quadratic equations. However in this case, the size of the system (even for one round) will be so huge that there will be no hope for any practical attacks.

B.6 Practical Consequences for Rijndael and AES

In Section 5.2 we tried to apply the XL algorithm, exactly as described in Appendix D.2 or in the paper [22]. It fails and there is no efficient algorithms known to solve such **general** systems of equations as above. However the systems obtained as described above are sparse. We consider for example the MQ problem we wrote for 128-bit Rijndael. For a general system of quadratic $R = 8000$ equations with $n = 1600$ variables, we have about $T = n^2/2 \approx 2^{20}$

[8] This is not certain, it is possible that XL is subexponential only on average, and AES gives some very special systems. Still it seems very likely to be true.

terms. This gives $R/T \approx 2^{-7.3}$ that is very small and the XL algorithm has to do extensive work in order to achieve $R'/T' \approx 1$, see Appendix D.2. In the MQ system we wrote above, it is easy to see that the number of terms is only about $T \approx (8*32+8*32+8+32+8)*(N_r*4*Nb)$. This gives only $R/T \approx 2^{-3.5}$ and suggests that for this system there **must** be a better method than XL. In Section 6.2 we will write such a system of quadratic equations in a different way in order to achieve an even higher value of R/T. For this there will be one variable for each input and each output bit of an S-box, which on one side leads to more equations and more (redundant) variables, but on the other side the system becomes more sparse.

C Simulations on XSL

The XSL attack is heuristic and in order to verify if it works as expected, one should do computer simulations. It is impossible to do it directly on Rijndael or Serpent, the systems are too big. Even if we restrict to Rijndael or Serpent to one round, the system will still be very big. Therefore we did some simulations on a smaller "toy ciphers". The goal is not prove that the XSL attack works for Rijndael but to see whether it behaves as predicted on small examples.
To know what is the exact complexity of the XSL attack for this or other concrete cipher, is a different (and more complex) question that requires even more simulations. Moreover there are many possible variants of XSL that might give very different results.

C.1 Simulations on a Toy Cipher

We build a toy cipher in the following way:

1. It is very similar to Serpent, except that the key schedule will just use permutations of bits, as in DES.
2. We will use mainly the notations from Section 2.1.
3. The size of the cipher will be small so that the attacks will be practical.
4. The S-box is the following permutation on $s = 3$ bits that has been chosen as a random non-linear permutation: $\{7, 6, 0, 4, 2, 5, 1, 3\}$.
5. It gives $r = 14$ fully quadratic equations with $t = 22$ terms, i.e. equations of the type:

$$\sum \alpha_{ij} x_i x_j + \sum \beta_{ij} y_i y_j + \sum \gamma_{ij} x_i y_j + \sum \delta_i x_i + \sum \epsilon_i y_i + \eta = 0$$

6. These equations are:

$$\begin{cases} 0 = x_1x_2 + y_1 + x_3 + x_2 + x_1 + 1 \\ 0 = x_1x_3 + y_2 + x_2 + 1 \\ 0 = x_1y_1 + y_2 + x_2 + 1 \\ 0 = x_1y_2 + y_2 + y_1 + x_3 \\ 0 = x_2x_3 + y_3 + y_2 + y_1 + x_2 + x_1 + 1 \\ 0 = x_2y_1 + y_3 + y_2 + y_1 + x_2 + x_1 + 1 \\ 0 = x_2y_2 + x_1y_3 + x_1 \\ 0 = x_2y_3 + x_1y_3 + y_1 + x_3 + x_2 + 1 \\ 0 = x_3y_1 + x_1y_3 + y_3 + y_1 \\ 0 = x_3y_2 + y_3 + y_1 + x_3 + x_1 \\ 0 = x_3y_3 + x_1y_3 + y_2 + x_2 + x_1 + 1 \\ 0 = y_1y_2 + y_3 + x_1 \\ 0 = y_1y_3 + y_3 + y_2 + x_2 + x_1 + 1 \\ 0 = y_2y_3 + y_3 + y_2 + y_1 + x_3 + x_1 \end{cases}$$

7. The number of rounds is N_r.
8. Let B be the number of S-boxes in each round. There are $B * s$ bits in each round, for convenience there are numbered here $0..Bs - 1$.
9. We will use a key of the same length: $H_k = B * s$ bits, so that one known plaintext will be (on average) sufficient to determine the key $K_0 = (K_{0\ 1}, \ldots, K_{0\ Bs})$ and therefore $\Lambda = 1$.
10. Each round i consists of the XOR with the derived key K_{i-1}, a parallel application of the B S-boxes, and then of a permutation of wires is applied.
    For the last round an additional derived key K_{N_r} is XORed.
11. Thus the linear equations from the key schedule will be (following the notations of Section 2.1) as follows:

$$X_{i+1\ j} = Z_{i\ j} \oplus [K_{i\ j}] \quad \text{for all } i = 0..N_r. \tag{4}$$

12. As in Section 2.1, we denote the plaintext by Z_0 and the ciphertext by X_{N_r+1}: they are considered as abbreviations for constants, not as variables.
13. The permutation of wires is defined as $j \mapsto (j+4 \bmod Bs)$, in other words the following equations are true:

$$Y_{i\ (j-4 \bmod Bs)} = Z_{i\ j} \quad \text{for all } i = 1..N_r. \tag{5}$$

14. The derived key K_i is obtained from K_0 by a permutation of wires:

$$[K_{i\ j}] \stackrel{def}{=} K_{0\ (j+i \bmod Bs)}.$$

15. There is no S-boxes in the key schedule, $D = 0$.
16. On this cipher (that resembles Serpent) we will apply a specific version of the second XSL attack described in Section 7.
17. We use the optimistic evaluation of P equal to $P = \lceil 22/14 \rceil = 2$.
18. Since $D = 0$, following Section 7.1 we will use one "artificial" S-box that contains all the key variables, and thus $E = 1$.
19. As in Section 7 we have $S = \Lambda * B * N_r + D + E = B * N_r + 1$.
20. The equations counted in R are: the initial $(S-E)*r$ equations multiplied by another equation form a different S-box, plus each of these equations multiplied by one of some t terms for some other "passive" S-box, plus each of these equations multiplied by one of H_k key variables. Following Section 6.3, we will replace t by $(t-r)$ in our computations. Thus we obtain:

$$R = r(S-E) * r(S-E-1)/2 + r(S-E) * (t-r) * (S-E-1) + r(S-E) * H_k.$$

    In practice we observed that for an unknown reason, if the $(t-r)$ terms are chosen in a certain way, the rank obtained will slightly decrease. Therefore, in order to obtain the best results we included al the possible equations (multiplying by all possible t terms) and only at a later stage we reduce their number by taking a random subspace of the space generated by these equations.
21. The equations on the diffusion part will be written on the basis of the equations from (4) $X_{i+1\ j} = Z_{i\ j} \oplus [K_{i\ j}]$ for all $i = 0..N_r$ in which for $i = 0$ the value $Z_{i\ j}$ will be replaced by the appropriate plaintext bit, and for $i \neq 0$ we replace it by $Y_{i\ (j-4 \bmod Bs)} = Z_{i\ j}$ from (5). There are $(N_r+1) * B * s$ such equations.
22. The equations counted in R' are: The equations above themselves, plus each of these equations multiplied by the H_k variables that is already present in the equation, plus each of them multiplied by one non-contant term for some S-box, with exclusion of some terms for the S-boxes that are connected with the current equation (but some products are still OK and does not increase the number of terms T in the attack). In the table below we will give the exact number R' examining all the possibilities one by one, here we give only an approximation:

$$R' \approx (N_r+1) * B * s + (N_r+1) * B * s + (N_r+1) * B * s(S-E) * (t-r).$$

    Here again, following Section 6.3 and Section 6.4, we replaced t by $(t-r)$ in our computations. In practice we generated all the equations. It is however important to compute the values R and R' as explained above in order to see if the number $Free$ of linearly independent equations is well (or not) approximated by $R+R'$. We will se that the answer is yes, and it suggests that the estimations of the complexity of the XSL attack given in Section 6.5 are close to reality.

23. The number of terms that appear in our equations include all the $t(S-E)+H_k$ initial terms and all products of terms from different S-boxes. This gives:

$$T = t(S-E) + H_k + t^2\binom{S-E}{2} + t(S-E) * H_k;$$

24. As we explained in Section 6.3 we will never achieve $\frac{Free}{T} > 1$. Following Section 6.1, our goal is to achieve $\frac{Free}{T-T'} > 1$.
25. Anyone can verify our simulations with any computer algebra system capable of reading and simple gaussian elimination on multivariate equations. We generated two concrete examples of the equations we used in the simulations for $N_r = 2$ and $N_r = 10$. They can be downloaded at: `http://www.minrank.org/example_xsl_2_2.zip`
and `http://www.minrank.org/example_xsl_2_10.zip`.
These two examples also contain detailed comments[9] and an exhaustive list of all terms with indication which of them are in T'.

In the tables below we present the results of the simulations.

S-box				Bs		H_k								The results	
s	r	t	B	[bits]	N_r	[bits]	Λ	S	R	R'	T	T'	$Free$	$\frac{Free}{T}$	$\frac{Free}{T-T'}$
3	14	22	2	6	1	6	1	3	588	284	742	336	727	0.9798	1.7906
3	14	22	2	6	2	6	1	5	2856	616	3241	840	3187	0.9833	1.3274
3	14	22	2	6	3	6	1	7	6804	1140	7504	1344	7329	0.9767	1.1273
3	14	22	2	6	4	6	1	9	12432	1856	13531	1848	13170	0.9732	1.1881
3	14	22	2	6	5	6	1	11	19740	2764	21322	2352	20711	0.9713	1.0918
3	14	22	2	6	6	6	1	13	28728	3864	30877	2856	29952	0.9700	1.0689
3	14	22	2	6	7	6	1	15	39396	5156	42196	3360	40893	0.9691	1.0530
3	14	22	2	6	8	6	1	17	51744	6640	55279	3864	53534	0.9684	1.0412
3	14	22	2	6	9	6	1	19	65772	8316	70126	4368	67875	0.9679	1.0322
3	14	22	2	6	10	6	1	21	81480	10184	86737	4872	83914	0.9675	1.0250
3	14	22	2	6	11	6	1	23	98868	12244	105112	5376	101654	0.9671	1.0192
3	14	22	2	6	12	6	1	25	117936	14496	125251	5880	121098	0.9668	1.0145
3	14	22	2	6	13	6	1	27	138684	16940	147154	6384	142235	0.9666	1.0104
3	14	22	2	6	14	6	1	29	161112	19576	170821	6888	165080	0.9664	1.0070
3	14	22	2	6	15	6	1	31	185220	22404	196252	7392	189621	0.9662	1.0040
3	14	22	2	6	16	6	1	33	211008	25424	223447	7896	215862	0.9661	1.0014
3	14	22	2	6	17	6	1	35	238476	28636	252406	8400	243803	0.9659	0.9992
3	14	22	2	6	18	6	1	37	267624	32040	283129	8904	273444	0.9658	0.9972
3	14	22	2	6	19	6	1	39	298452	35636	315616	9408	304785	0.9657	0.9954
3	14	22	2	6	20	6	1	41	330960	39424	349867	9912	337826	0.9656	0.9937
3	14	22	2	6	21	6	1	43	365148	43404	385882	10416	372567	0.9655	0.9923
3	14	22	2	6	22	6	1	45	401016	47576	423661	10920	409008	0.9654	0.9910
3	14	22	2	6	23	6	1	47	438564	51940	463204	11424	447149	0.9653	0.9897
3	14	22	2	6	24	6	1	49	477792	56496	504511	11928	486990	0.9653	0.9886
3	14	22	2	6	25	6	1	51	518700	61244	547582	12432	528531	0.9652	0.9876

We see that that when $B = 2$, the XSL attack works for up to 16 rounds.

[9] Text and/or equations after a ";" should be ignored

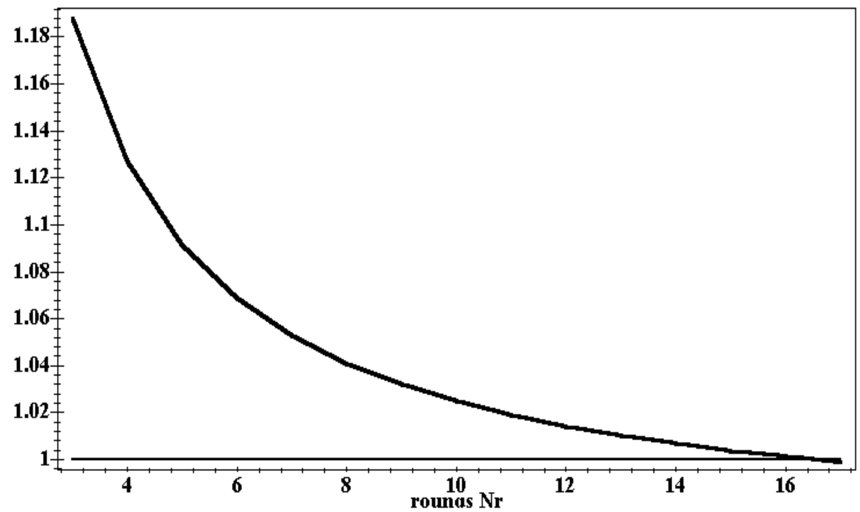

Fig. 1. The value $\frac{Free}{T-T'}$ as a function of the number of rounds N_r.

Here is another series of simulations with $B = 4$ and $B = 8$.

S-box				Bs		H_k								The results	
s	r	t	B	[bits]	N_r	[bits]	Λ	S	R	R'	T	T'	$Free$	$\frac{Free}{T}$	$\frac{Free}{T-T'}$
3	14	22	4	12	1	12	1	5	3192	952	3751	912	3693	0.9845	1.3008
3	14	22	4	12	2	12	1	9	13104	2384	14545	1920	14184	0.9752	1.1235
3	14	22	4	12	3	12	1	13	29736	4584	32395	2928	31470	0.9714	1.0680
3	14	22	4	12	4	12	1	17	53088	7552	57301	3936	55556	0.9695	1.0411
3	14	22	4	12	5	12	1	21	83160	11288	89263	4944	86442	0.9684	1.0252
3	14	22	4	12	6	12	1	25	119952	15792	128281	5952	124128	0.9676	1.0147
3	14	22	4	12	7	12	1	29	163464	21064	174355	6960	168614	0.9670	1.0073
3	14	22	4	12	8	12	1	33	213696	27104	227485	7968	219900	0.9667	1.0017
3	14	22	4	12	9	12	1	37	270648	33912	287671	8976	277986	0.9663	0.9975
3	14	22	4	12	10	12	1	41	334320	41488	354913	9984	342872	0.9661	0.9940
3	14	22	4	12	11	12	1	45	404712	49832	429211	10992	414558	0.9659	0.9912
3	14	22	4	12	12	12	1	49	481824	58944	510565	12000	493044	0.9657	0.9889

S-box				Bs		H_k								The results	
s	r	t	B	[bits]	N_r	[bits]	Λ	S	R	R'	T	T'	$Free$	$\frac{Free}{T}$	$\frac{Free}{T-T'}$
3	14	22	8	24	1	12	1	9	14448	3440	16573	2064	16212	0.9782	1.1174
3	14	22	8	24	2	12	1	17	55776	9376	61345	4080	59600	0.9716	1.0408
3	14	22	8	24	3	12	1	25	123984	18384	134431	6096	130188	0.9691	1.0110
3	14	22	8	24	4	12	1	33	316736	30464	235561	8112	227796	0.9678	1.0023

C.2 Conclusion

Apparently both $\frac{Free}{T}$ and $\frac{Free}{T-T'}$ either converge to a fixed value, or they decrease very slowly. If they converge, both limits are identical, because it can be shown that $\frac{T-T'}{T} \to 1$. Surprisingly it seems that this limit is the same for $B = 2$, $B = 4$ and $B = 8$.
We see that for $B = 2$ the XSL attack will work for up to 16 rounds. Only for 17 rounds we have $\frac{Free}{T-T'} < 1$. A similar behaviour is observed when $B = 4$ and when $B = 8$.
When $\frac{Free}{T-T'} < 1$, there are probably some ways to improve the attack. Ultimately, since we observe that $\frac{Free}{T}$ seems to converge to a limit that is below 1, and since $\frac{T'}{T} \to 0$, starting from some number of rounds it will be necessary to increase P to make the attack work.
More simulations and/or a better theory to understand the behaviour of the XSL attack for bigger ciphers and for more rounds.

D The XL Algorithm

In order to make this paper self-sufficient we describe the XL algorithm for the case of $GF(2)$. We also recall the simplified analysis of the complexity of XL from [22], that seems approximatively correct. For experimental results on XL one should refer to section D.7 or to the original paper [22].

D.1 Solving MQ with the XL Algorithm

The origin of the XL algorithm was the relinearization algorithm presented by Shamir and Kipnis at Crypto'99. From the relinearization algorithm, it seemed obvious that if the system of equations is overdefined, then the problem is much easier. In a paper published at Eurocrypt'00 [22], authors propose a new algorithm called XL, that can be seen as an improved version of relinearization.

D.2 How XL Works

We consider the problem of solving m quadratic equations with n variables that are in $GF(2)$. In general, the number of quadratic terms in these equations is about $t \approx n^2/2$ (but it can be less).
Let $D = 2, 3, \ldots$ be a parameter of the XL algorithm. What the algorithm basically does, is to multiply each possible equation $1...m$ by all possible products of $D-2$ variables. Thus we get about: $R \approx \binom{n}{D-2} m$ new equations. The total number of terms that appear in these equations is about $T = \binom{n}{D}$. We expect that most of the equations are linearly independent. Then, we pick a sufficiently big D such that

$$R = \binom{n}{D-2} m \geq \binom{n}{D} = T.$$

Obviously the number of linearly independent equations cannot exceed the number of terms T. We expect that if the system has a unique solution (see Section D.4), then there is such a D for which $R \geq T$, and such that also the number $Free$ of linearly independent equations in R will be very close to T. Then if the rank deficit $T - Free$ is not too big, we expect that the system will be solved. It is easy when $T - Free$ is a very small number, but still possible when $T - Free$ is quite big. For example let T' be the number of terms out of T that contain only the first 40 variables. If $Free > T - T' + 40$, then we are able to obtain (by progressive elimination of terms) to obtain a system of 40 equations with 40 variables that can be solved by the exhaustive search. Then we fix these 40 variables and we should obtain $T - Free$ much smaller in the new system, and it will probably not be necessary to repeat the above "trick" with some other 40 variables.
We expect that the D value for which XL works is equal or very close to the theoretical value D for which $R \geq T$. Thus the XL algorithm is expected to succeed when:

$$R \geq T \quad \Rightarrow \quad m \geq \binom{n}{D} / \binom{n}{D-2} \approx n^2/D^2.$$

This gives

$$D \approx \frac{n}{\sqrt{m}}$$

and the complexity of the attack is about

$$T^\omega \approx \binom{n}{D}^\omega \approx \binom{n}{n/\sqrt{m}}^\omega$$

with $\omega \leq 3$ being the exponent of the Gaussian reduction. It is unclear what value ω will be realistic in our attacks, see Section F.
From the above formula it seems that XL is subexponential, however very little is known about the actual behaviour of XL for very big systems of equations.

D.3 Remarks by T.T. Moh on XL

In [13] T.T. Moh states that "From the theory of Hilbert-Serre, we may deduce that the XL program will work for many interesting cases for D large enough".
In Section 4 of [13] the author shows a very special example on which XL fails, however he did not fully understand the power of XL, for example with FXL, or an appropriate final step with T' such as described above in Section D.2, or the version described in Section 6.1.
In Section 3 the author makes a serious mistake. He assumes $D \gg n$ in a formula in which $D = \mathcal{O}(\frac{n}{\sqrt{m}})$. He shows that $Free/R \approx \frac{(n+D)(n+D-1)}{D(D-1)m} = w$ and obviously $w \to \frac{1}{m}$ when $D \to \infty$. However D is never as big as n, if we assume that we have $D \approx \frac{n}{\sqrt{m}}$ as in the previous section, we get $w \approx 1$. The conclusion of T.T. Moh is incorrect.

D.4 Unicity of the Solution

In the paper [22], authors made many computer simulations on XL algorithm in the field $GF(127)$. In some cases XL failed, and this is apparently due to the fact that the system had many solutions, not in the base field $GF(127)$, but is some algebraic extension. Indeed such manipulations on the equations that are done in XL (described above): multiplying equations by monomials and combining them, conserve all the solutions in the algebraic closure of $GF(127)$. This is not a problem for small fields, for example GF(2). When multiplying such equations by monomials of a small degree, we will make explicit usage of the equation of the field $x_i^2 = x_i$ for each of the variable x_i, and always write x_i instead of x_i^2. Such repeated interaction with the equation of the field will eliminate all the solutions with variables being not in $GF(2)$.
Another problem with XL is that if there are many solutions, there is no simple algebraic equation that would englobe all of them, and the algorithm has to fail.
Conversely it seems that for a system of quadratic equations over a small field $GF(2)$(and also other $GF(q)$ with q small), that has only one solution in the base field $GF(2)$, the XL method will always work, except maybe for some very special systems.

D.5 XL and Sparsity

It is obvious that if in the initial system $t < n^2/2$, i.e. not all possible $n^2/2$ quadratic terms are present, XL will work better. After multiplying each of the equations $\binom{n}{D-2}$ by one of the terms, it may happen that not all the possible $\binom{n}{D}$ terms will be obtained. In this case we might obtain a strictly smaller D, for which the number of linearly independent equations will be big enough. Since the algorithm is exponential in D, lowering it even by one, will yield a dramatic improvement in the complexity. This improvement will be even better if the terms have some specific structure that will allow us to multiply them by only some selected monomials. This should be done in such a way that, as much as possible different products of some monomial with some of the initial terms (i.e. present in the initial equations), should lead to identical terms of degree D. Thus we will generate many equations while maintaining the total number of terms small. The XSL attack we introduce in Section 6, has been designed in such a way.

D.6 Does XL Always Work ?

It is important to understand that the XL algorithm will not always work. Following the XL complexity evaluation, an overdefined system of equations (big m/n) leads to a dramatic improvement in XL complexity compared to other systems with the same number of variables (the case of underdefined MQ is studied in [5]). Still it is easy to produce overdefined systems on which it fails. For example if we mix two systems of equations with separate sets of variables, one of which is very much overdefined, and the other of which is not, we will still obtain a largely overdefined system of equations. However applying XL will only find solutions to one of the systems, and never to the other.

Bad things may also happen when variables are linearly dependent. For example consider a system of $m = 100$ equations with $n = 100$ variables over $GF(2)$. If we apply XL to this system we have: $D \approx n/\sqrt{m} \approx 10$ and the complexity of the XL attack is very big: about $\binom{n}{D}^{2.376} \approx 2^{104}$. Now we add just 10 additional variables that are linear combinations of the existing variables. It allows to write 10 new linear equations and to derive $10 * n = 10 * 100$ new quadratic equations. Everything seems correct: all these equations will be linearly independent. Now we have a new system of $m' = 1110$ quadratic equations with $n' = 110$ variables. If we naively apply XL, we get $D' \approx n'/\sqrt{m'} \approx 4$ and the complexity of the XL attack would be only: $\binom{n'}{D'}^{2.376} \approx 2^{53}$. It is less than before, though our system is just the expansion of the previous system. In reality, the XL algorithm will certainly fail for this second (very special) system. The exact analysis of the complexity of XL for systems having dependent variables is not as simple anymore. For example in the relinearization technique from [21, 22], when some variables are products of some other variables, less linearly independent equations than expected are obtained, see [22]. The relinearization algorithm still works, but not as well as XL: it seems that adding new variables that are defined as combinations of the previous variables is a bad idea. It will create more than expected linear dependencies at some further stage, see [22].

There are many questions open about XL and similar methods. In general we tend to believe that, if such methods doesn't not work, there is usually a combinatorial or algebraical reason for this, and sooner or later we will find out how to prove that it does not work. Currently it seems that (at least) these conditions should be satisfied for methods such as XL, XSL (or relinearization) to work:

1. The system should have a unique solution.
2. The variables should be "well mixed".
3. There shouldn't be possible to exhibit a subsystem and a variable change, for which the subsystem contains less terms than the expected contribution from this subsystem, to the total number of linearly independent equations.

On the other side, if we are not able to prove that the attack fails, one should assume that it may (or may not) work and should do computer simulations, that would either invalidate the claim, either give a partial confirmation. This is the approach of [22] and of Section C.

D.7 Simulations on XL

In the paper that describes XL, the authors demonstrate that XL works with a series of computer simulations over $GF(127)$ (some more are given in the extended version of the paper [22]). Since then, T.T.Moh makes in [13] some reserves whether the XL algorithm actually works as expected. See Section D.3 to see why these remarks are unsubstantial. In this section we present some computer simulations on the XL algorithm over $GF(2)$. No such simulations have been published so far.

In all the simulations that follow we will pick a random system of linearly independent quadratic non-homogenous equations $y_i = f_i(x_1, \ldots, x_n)$ and pick a random input $x = (x_1, \ldots, x_n)$. Then we modify the constants in the system, in order to have a system that gives 0 in x, i.e. we write a system to solve as $\forall\, i \quad l_i(x_1, \ldots, x_n) = 0$. If n is not too big, we also require that the system has a unique solution, which is the case with good probability.
In the following table we fix n and try a random system of m linearly independent equations with growing m and with a fixed D. We denote by R the number of equations generated, T is the number of terms $T \approx \frac{n}{D}$. $Free$ is the number of linearly independent equations and T' is the number of terms that can be multiplied by one variable, for exemple x_1. The attack is expected to work when $Free/(T - T') > 1$, see Sections D.2 and 6.1.

n	10	10	10	10	10	10	10	10	10
m	10	11	12	13	14	15	16	17	18
D	3	3	3	3	3	3	3	3	3
R	110	121	132	143	154	165	176	187	198
T	176	176	176	176	176	176	176	176	176
T'	92	92	92	92	92	92	92	92	92
$Free$	110	121	132	143	154	165	174	175	175
$\frac{Free}{R}$	1.000	1.000	1.000	1.000	1.000	1.000	.9886	.9358	.8838
$\frac{Free}{T}$	.6250	.6875	.7500	.8125	.8750	.9375	.9886	.9943	.9943
$\frac{Free}{T-T'}$	1.310	1.441	1.571	1.702	1.833	1.964	2.071	2.083	2.083

n	10	10
m	10	11
D	4	4
R	560	616
T	386	386
T'	260	260
$Free$	385	385
$\frac{Free}{R}$	.6250	.6875
$\frac{Free}{T}$	.9974	.9974
$\frac{Free}{T-T'}$	3.056	3.056

n	20	20	20	20	20	20	20	20	20
m	20	22	24	26	28	30	32	34	36
D	4	4	4	4	4	4	4	4	4
R	4220	4642	5064	5486	5908	6330	6752	7174	7596
T	6196	6196	6196	6196	6196	6196	6196	6196	6196
T'	2320	2320	2320	2320	2320	2320	2320	2320	2320
$Free$	4010	4389	4764	5135	5502	5865	6195	6195	6195
$\frac{Free}{R}$	.9502	.9455	.9408	.9360	.9313	.9265	.9175	.8635	.8156
$\frac{Free}{T}$	.6472	.7084	.7689	.8288	.8880	.9466	.9998	.9998	.9998
$\frac{Free}{T-T'}$	1.035	1.132	1.229	1.325	1.420	1.513	1.598	1.598	1.598

n	20	20
m	20	22
D	5	5
R	27020	29722
T	21700	21700
T'	10072	10072
$Free$	21699	21699
$\frac{Free}{R}$	.8031	.7301
$\frac{Free}{T}$	1.000	1.000
$\frac{Free}{T-T'}$	1.866	1.866

E A Toy Example for the "T' method"

This is a concrete working example for the final step of the XSL algorithm called the "T' method". It can also be applied to the XL algorithm.

We have $n = 5$ variables, and thus $T = 16$ and $T' = 10$. We start with a random system having exactly one solution, and with $Free > T - T'$ and with 2 exceeding equations, i.e. $Free = T - T' + 2$. Here is a system in which T' is defined with respect to x_1:

$$\begin{cases} x_3x_2 = x_1x_3 + x_2 \\ x_3x_4 = x_1x_4 + x_1x_5 + x_5 \\ x_3x_5 = x_1x_5 + x_4 + 1 \\ x_2x_4 = x_1x_3 + x_1x_5 + 1 \\ x_2x_5 = x_1x_3 + x_1x_2 + x_3 + x_4 \\ x_4x_5 = x_1x_2 + x_1x_5 + x_2 + 1 \\ 0 = x_1x_3 + x_1x_4 + x_1 + x_5 \\ 1 = x_1x_4 + x_1x_5 + x_1 + x_5 \end{cases}$$

Here is the same system in which T' is defined with respect to x_2:

$$\begin{cases} x_1x_3 = x_3x_2 + x_2 \\ x_1x_4 = x_3x_2 + x_2 + x_1 + x_5 \\ x_1x_5 = x_2x_4 + x_3x_2 + x_2 + 1 \\ x_3x_5 = x_2x_4 + x_3x_2 + x_2 + 1 + x_4 + 1 \\ x_3x_4 = x_2x_4 + x_1 + 1 \\ x_4x_5 = x_1x_2 + x_2x_4 + x_3x_2 \\ 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \end{cases}$$

We have $rank = 8$. Now multiply the two exceeding equations of the first version of the system by x_1.

$$\begin{cases} 0 = x_1x_3 + x_1x_4 + x_1 + x_1x_5 \\ 0 = x_1x_4 \end{cases}$$

We have $rank = 10$. We get two new linearly independent equations.

We rewrite these equations, using the second system, only with terms that can be multiplied by x_2. Now we have 4 exceeding equations for the second system (two old and two new):

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5 + x_2 + 1 \\ 0 = x_3x_2 + x_2 + x_1 + x_5 \end{cases}$$

We multiply these four equations by x_2.

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_2x_4 + x_2 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5x_2 \\ 0 = x_3x_2 + x_2 + x_1x_2 + x_2x_5 \end{cases}$$

We are not lucky, the second equation is invariant by this transformation. Still we get three new linearly independent equations. We have $rank = 13$.

We rewrite, using the first system, the three new equations with terms that can be multiplied by x_1.

$$\begin{cases} 1 = x_1x_5 + x_2 + x_3 + x_4 \\ 1 = x_1x_2 + x_1x_3 + x_1x_5 + x_2 + x_3 + x_4 \\ 0 = x_3 + x_4 \end{cases}$$

Still $rank = 13$. Then we multiply the three new equations by x_1:

$$\begin{cases} 1 = x_1x_5 + x_1x_2 + x_1x_3 + x_1x_4 \\ 1 = x_1x_5 + x_1x_4 \\ 0 = x_3 + x_4 \end{cases}$$

We have $rank = 14$. We get one more linearly independent equation. The two other are redundant. Now we rewrite the first equation with terms that can be multiplied by x_2:

$$0 = x_1x_2 + x_2x_4 + x_3x_2 + x_1 + x_2 + x_5$$

We have still $rank = 14$. Then we multiply the new equation by x_2:

$$0 = x_2x_4 + x_3x_2 + x_2x_5 + x_2$$

We get another new linearly independent equation. We have $rank = 15$. The rank is the maximum that can be achieved, there are 15 non-zero monomials here, and $rank = 16$ can only be achieved for a system that is contradictory.

F About the Value of ω

F.1 What is the Complexity of Gaussian Reduction ?

In practice it is usually assumed that $\omega = 3$. We prefer to use a fairly theoretical result on the best known exponent for the Gaussian reduction from the paper [3], that shows that $\omega \leq 2.376$. The (neglected) constant factor in this algorithm is unknown to the authors of [3], and is expected to be very big. Still, **we claim that in cryptography one should be optimistic on attacks, in order not to be surprised by the future improvements**. In this paper we deal with extremely big systems of equations, and therefore even a big constant will be relatively small. For other reasons, even a constant as big as 20000, can certainly be neglected. This is because we need to have a fair measure of complexity compared to the exhaustive search. In the exhaustive search, the unitary operation is one encryption, that will take for example about 300 CPU cycles. For our attacks, unitary operation is the addition of bits modulo 2, and it is possible to do about 64 such binary additions modulo 2 in parallel in one single CPU clock. Therefore the unit is about $64 * 300 \approx 20000$ times smaller.

F.2 Further Improvements, or Can ω be Even Less in XSL ?

There are some hopes to achieve a further improvement in ω. On one hand this might come from new algorithms for Gaussian reduction being discovered, which seems to stumble on some difficult computational problems, see [19, 3].

On the other hand, it is very likely that the elimination can be done faster in the special case of systems generated in the XSL attack. Clearly the final (big) system is still quite **sparse** and have a **very regular** structure. For example it is possible to compute **in constant time** a list of all equations that contain a given term. Therefore it is probably possible to design a progressive elimination technique. Such a technique would, instead of generating a huge system of equations and eliminating all terms in it, generate the system by parts and eliminate terms for smaller systems, in such a (clever) way that the terms that have already been eliminated will not be generated anymore. It could also use special data structure that is dynamically updated with a reasonable cost, in order to be able to always find all the equations that contain a given term in sub-quadratic (or maybe even linear) time, i.e. faster than in the general case.

It is unclear how much can be gained from a careful combination of all these ideas. It seems not completely unsound to believe that the complexity might be reduced even to $\mathcal{O}(T^2)$, i.e. ω might be as low as 2.

Remark: Efficient methods for solving big systems of multivariate quadratic equations already exist and are based on Gröbner bases. Thus for example in [6] it is shown how to find a solution to the HFE Challenge 1 [18] in 2^{62} using 390 Giga-bytes of disk space. On April 10th 2002, at the cryptographic seminar at Versailles University, Jean-Charles Faugère from Paris 6 University have presented an implementation of his recent Gröbner bases algorithm F5/2 that managed to solve the same HFE challenge 1 in 96 hours on an 833 MHz Alpha workstation with 4 Gigabytes of memory. It seems that the equations discovered in [6] are precisely the same that allow the F5/2 algorithm to work efficiently. From this, we expect that the F5/2 algorithm will also help to solve the equations obtained in the XSL attacks much faster than expected.