Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt

Nicolas T. Courtois

Cryptography research, Schlumberger Smart Cards,
36-38 rue de la Princesse, BP 45, 78430 Louveciennes Cedex, France
http://www.nicolascourtois.net
courtois@minrank.org

Abstract. Many stream ciphers are built of a linear sequence generator and a non-linear output function f. There is an abundant literature on (fast) correlation attacks, that use linear approximations of f to attack the cipher. In this paper we explore higher degree approximations, much less studied. We reduce the cryptanalysis of a stream cipher to solving a system of multivariate equations that is overdefined (much more equations than unknowns). We adapt the XL method, introduced at Eurocrypt 2000 for overdefined quadratic systems, to solving equations of higher degree. Though the exact complexity of XL remains an open problem, there is no doubt that it works perfectly well for such largely overdefined systems as ours, and we confirm this by computer simulations. We show that using XL, it is possible to break stream ciphers that were known to be immune to all previously known attacks. For example, we cryptanalyse the stream cipher Toyocrypt accepted to the second phase of the Japanese government Cryptrec program. Our best attack on Toyocrypt takes 2^{92} CPU clocks for a 128-bit cipher. The interesting feature of our XL-based higher degree correlation attacks is, their very loose requirements on the known keystream needed. For example they may work knowing ONLY that the ciphertext is in English.

Key Words: Algebraic cryptanalysis, multivariate equations, overdefined equations, Reed-Muller codes, correlation immunity, XL algorithm, Gröbner bases, stream ciphers, pseudo-random generators, nonlinear filtering, ciphertext-only attacks, Toyocrypt, Cryptrec.

1 Introduction

The security of most cryptographic schemes is usually based on impossibility to extract some secret information, given access to some encryption, signature oracles or other derived information. In most useful cases, there is no security in information-theoretic setting: the adversary has usually enough information to uniquely determine the secret (or the ability) he wants to acquire. Moreover the basic problem is always (in a sense) overdefined: the adversary is assumed to have at his disposal, for example, great many plaintext and cipher text pairs, message and signature pairs, etc. He usually has available, much more than the information needed to just determine the secret key. Thus, one might say, most cryptographic security relies on the hardness of largely overdefined problems. In public key cryptography, the problem is addressed by provable security, that will assure that each utilization of the cryptographic scheme does not leak useful information. The security is guaranteed by a hardness of a single difficult problem, and will not degrade with the repetitive use of the scheme. However unfortunately, there is yet very little provable security in secret key cryptography. It is also in secret key cryptography that the problems become most overdefined, due to the amounts of data that are usually encrypted with one single session key. This is especially true for stream ciphers: designed to be extremely fast in hardware, they can encrypt astronomic quantities of data, for example on an optical fiber.

In this paper we consider a popular class of stream ciphers, in which there is a linear part, producing a sequence with a large period (for example one or several LFSRs), and a stateless nonlinear part that produces the output, given the state of the linear part. The security of such stream ciphers have been studied by many authors. In [11], Golic gives a set of criteria that should be satisfied in order to resist to the known attacks on stream ciphers. For example, a stream cipher should resist to the fast correlation attack [15], the conditional correlation attack [1] and the inversion attack [11]. In order to resist different types of correlation attacks, many authors focused on proposing boolean functions that will have no good linear approximation and that will be correlation immune with regard to a subset of several input bits, see for example [3]. In this paper we will exploit rather correlation properties with regard to a non-linear low degree multivariate functions that use all of the variables, or in other words, low degree approximations. This kind of correlations has already been studied, see for example in [12, 9] but it seems that their application to cryptographic attacks did not receive sufficient attention.

In this paper we explain that many constructions of stream ciphers directly give an overdefined system of multivariate equations of low degree. Many other can approximated with a good probability by such a system of equations. All this is not completely new, however only recently people became aware of the existence of efficient algorithms for solving some systems of multivariate equations of low degree, see [24, 6, 7]. The XL algorithm, proposed by Courtois, Klimov, Patarin and Shamir [24], allows to solve overdefined systems of equations much faster than expected. Later, Courtois and Pieprzyk proposed the XSL method [7], to further improve XL when the equations are sparse. In the same paper [7], authors propose to use multivariate polynomial equations to attack block ciphers such as AES. Unfortunately, these attacks are, to say the least, heuristic. In this paper we study multivariate algebraic attacks on stream ciphers. Unlike in [7], our systems of equations will be much more overdefined. We show that in this case it is possible to predict the behaviour of the XL method with precision and confidence.

In this paper, using XL, we introduce a method to exploit (at least in theory) any correlation with a function of any degree, to mount an attack on a class of popular stream ciphers. Moreover we demonstrate that such attacks can be indeed faster than exhaustive search of the key for some real stream ciphers, for example for Toyocrypt.

The paper is organized as follows: in Section 2 and in Appendix A we study the XL algorithm from [24] for solving multivariate quadratic equations, and extend it to equations of higher degree. In Section 3 we apply XL to the cryptanalysis of stream ciphers. In Section 4 we discuss the opportunity to use bent functions in stream ciphers. Then in Section 5 we apply our attack to Toyocrypt stream cipher.

2 The XL Algorithm

In this paper we describe a rather obvious extension of the XL algorithm [24]. Instead of solving a system of m multivariate quadratic equations with n variables of degree $d = 2$ as in [24], we consider also higher degree equations, i.e. study the general case $d \geq 2$. Let D be the parameter of the XL algorithm. Let $l_i(x_0, \ldots, x_{n-1}) = 0$ be the initial m equations, $i = 1 \ldots m$ with n variables $x_i \in GF(2)$. The XL algorithm consists of multiplying both sides of these equations by products of variables:

Definition 2.0.1 (The XL algorithm). Execute the following steps:

1. **Multiply:** Generate all the products $\prod_{j=1}^{k} x_{i_j} \cdot l_i$ with $k \leq D - d$, so that the total degree in the x_i of these equations is $\leq D$.
2. **Linearize:** Consider each monomial in the x_i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in 1. The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
3. **Get a Simpler Equation:** Assume[1] that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite field (e.g., with Berlekamp's algorithm).
4. **Final step:** It should not be necessary to repeat the whole process. Once the value of x_1 is known, we expect that all the other variables will be obtained from the same linear system.

[1] Improved versions of the XL algorithm exist in which the system can still be solved even if this condition is not satisfied, see the FXL algorithm [24], and XL' and XL2 methods described in [6]. We do not need these improvements here.

We expect that to find one solution to the system, the complexity of XL will be essentially the complexity of one single Gaussian reduction in the step 2.

2.1 The Necessary Condition for XL to Work

The XL algorithm consists of multiplying the initial m equations l_i by all possible monomials of degree up to $D-d$, so that the total degree of resulting equations is D. Let R be the number of equations generated in XL, and T be the number of all monomials. We have, (the first term is dominant):

$$R = m \cdot \left(\sum_{i=0}^{D-d} \binom{n}{i} \right) \approx m \cdot \binom{n}{D-d}, \qquad T = \sum_{i=0}^{D} \binom{n}{i} \approx \binom{n}{D}$$

The main problem in the XL algorithm is that in practice not all the equations generated are independent. Let $Free$ be the exact number of equations that are linearly independent in XL. We have $Free \le R$. We also have necessarily $Free \le T$.

The main heuristics behind XL is the following: it can be seen that for some D we have always $R \ge T$. Then we expect that $Free \approx T$, as obviously it cannot be bigger than T. More precisely, following [24], when $Free \ge T - D$, it is possible by Gaussian elimination, to obtain one equation in only one variable, and XL will work. Otherwise, we need a bigger D, or an improved algorithm[2].

The Saturation Problem in XL

The exact value of $Free$ in XL is somewhat complex to predict. In [24] authors demonstrate that XL works with a series of computer simulations for $d = 2$ and over $GF(127)$. In [6] authors show that it also works very well for $d = 2$ and over $GF(2)$. Moreover they explain how to predict the exact number $Free$ of linearly independent equations in XL. In this paper we extend the study of XL for higher degree equations $d > 2$ (still over $GF(2)$), and will also give a formula that allows to compute $Free$ (see Conjecture A.3.1). This will allow us to say that our (cryptanalytic) applications of XL should work **exactly** as predicted.

In order to XL algorithm to work, it is sufficient that for some D, the number $Free$ of linearly independent equations satisfies $Free \ge T - D$. In [19], 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 Moh shows a very special example on which the basic version[2] of XL always fails, for any D [19]. This example is very

[2] Improved versions of XL exist [24, 6], see the footnote 1 on the previous page.

interesting, however it seems that such counter-example does not exist when XL is done over a small finite field, see [13] and [6].

Remark: In Section 3 of [19] Moh gives another misleading argument. He assumes $D \gg n$ in a formula in which $D = \mathcal{O}(\frac{n}{\sqrt{m}})$. He shows that, apparently $Free/R \approx \frac{(n+D)(n+D-1)}{D(D-1)m} = w$, and it is obvious that $w \to \frac{1}{m}$ when $D \to \infty$. However in XL, 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 Moh is inappropriate, not to say incorrect.

According to [13], when D is sufficiently big, we will always have $Free = T - \alpha$, with α being the number of solutions to the system[3]. Therefore for systems over GF(2) that have one and unique solution we expect to always achieve $Free = T - 1 > T - D$, which is called saturation[4]. In all our simulations we observed that this saturation, necessary to solve the system, is achieved very quickly, and in fact as soon as $R > T$.

2.2 Asymptotic Analysis of XL for Equations of Degree d

We assume that $D \ll n$. XL algorithm is expected to succeed when $R \geq T$, i.e. when

$$m\binom{n}{D-d} \geq \binom{n}{D} \quad \Rightarrow \quad m \geq \frac{(n-D+d)\cdots(n-D+1)}{D(D-1)\cdots(D-d+1)}$$

Thus (assuming that $D \ll n$) we get:

$$D \approx \frac{n}{m^{1/d}}, \quad \text{and} \quad T^{\omega} \;\approx\; \binom{n}{D}^{\omega} \;\approx\; \binom{n}{n\,/m^{1/d}}^{\omega}$$

Asymptotically this is expected to be a good evaluation, when $m = \varepsilon n^d$ with a constant $\varepsilon > 0$.

The Complexity of XL and Gaussian Reduction

Let ω be the exponent of the Gaussian reduction. In theory it is at most $\omega \leq 2.376$, see [4]. However the (neglected) constant factor in this algorithm is expected to be very big. The fastest practical algorithm we are aware of, is Strassen's algorithm that requires about $7 \cdot T^{log_2 7}$ operations. Since our basic operations are over $GF(2)$, we expect that a careful bitslice implementation of this algorithm on a modern CPU can handle 64 such operations in one single CPU clock. To summarize, we evaluate the complexity of the Gaussian reduction to be $7/64 \cdot T^{log_2 7}$ CPU clocks.

---

3 Here however one should include also the solutions at infinity. Such solutions do not exist when the equations of the field $x_i^2 = x_i$ are included in XL, see [6] and [13].

4 It is easy to show that $Free = T$ is impossible for a system that has a solution, and more generally if α is the number of solutions (including points at infinity, see the footnote 3), one always has $Free \leq T - \alpha$ in XL, cf. [13].

The Exact Behaviour of XL for Interesting Cases $\mathbf{d \geq 2}$. In this paper we do not use any of the estimations given above. We study the exact behaviour of XL, and compute the exact values of $Free$ for the interesting values of d and D. This part is in Appendix A.

3 Application of XL to Stream Ciphers

In this part we outline a general strategy to apply XL in cryptanalysis of a general class of stream ciphers. Later we will apply it to Toyocrypt.

3.1 The Stream Ciphers that May be Attacked

We consider only synchronous stream ciphers, in which each state is generated from the previous state independently of the plaintext, see for example [17]. We consider regularly clocked stream ciphers, and also (it makes no difference) stream ciphers that are clocked in a known way[5].

For simplicity we restrict to binary stream ciphers in which the state and keystream are composed of a sequence of bits b_i. Let L be the "connection function" that computes the next state. We restrict to the (very popular) case of cipher with linear feedback, i.e. when L is linear over $GF(2)$. We assume that L is public, and only the state is secret. We also assume that the function f that computes the output bit from the state is public and does not depend on the secret key of the cipher. The only no-linear component of the cipher is f and this way of building stream ciphers is sometimes called "nonlinear filtering". It includes the very popular filter generator, in which the state of a single LFSR[6] is transformed by a boolean function, and also not less popular scenarios, in which outputs of several LFSR are combined by a boolean function (combinatorial function generators or nonlinear function generators).

The problem of cryptanalysis of such a stream cipher can be described as follows. Let $(k_0, \ldots, k_{n-1})$ be the initial state, then the output of the cipher (i.e. the keystream) is given by:

$$\begin{cases} b_0 = f \quad\ (k_0, \ldots, k_{n-1}) \\ b_1 = f\left(L\ (k_0, \ldots, k_{n-1})\right) \\ b_2 = f\left(L^2(k_0, \ldots, k_{n-1})\right) \\ \qquad \vdots \end{cases}$$

The problem we consider[7] is to recover $(k_0, \ldots, k_{n-1})$ given some b_i.

[5] This condition can sometimes be relaxed, see the attacks on LILI-128 in [5].

[6] A Linear Feedback Shift Register, see for example [17]. It is also possible to use a Modular LFSR, i.e. a MLFSR, which is equivalent in theory, see, [18], but may be better in practice. A MLFSR is used in the Toyocrypt cipher that we study later.

[7] We do not consider attacks in which one can predict the future keystream, given some information on the current keystream, and without computing the key.

3.2 The Attack Scenario

We are going to design a partially known plaintext attack, i.e. we know some bits of the plaintext, and the corresponding ciphertext bits. These bits does not need to be consecutive. For example if the plaintext is written with latin alphabet and does not use too much special characters, it is very likely that all the characters have their most significant bit equal to 0. This will be enough for us, if the text is sufficiently long. In our later attacks we just assume that we have some m bits of the keystream at some known positions: $\{(t_1, b_{t_1}), \ldots, (t_m, b_{t_m})\}$.

Remark: Even if no bit of plaintext is known, there are many cases in which our attack can be extended. For example if the plaintext contains parity bits.

3.3 Criteria on the Function *f*

Let f be the boolean function[8] that is used to combine the bits of the linear part of a stream cipher (the entries of the function are for example some bits of the state of some LFSRs). There are many design criteria known on boolean functions. For example f should be balanced and have high algebraic degree. In order to resist different types of correlation attacks, many authors focused on proposing boolean functions that will have no good linear approximation and that will be correlation immune with regard to a subset of several input bits, see for example [3]. In this paper we will exploit rather correlation properties with regard to a non-linear low degree multivariate functions that use all of the variables, or in other words, low degree approximations. This kind of correlations has already been studied, see for example in [12, 9] but it seems that their application to cryptographic attacks did not receive sufficient attention.

Our attack works in two cases:

S1 When the boolean function f has a low algebraic degree d.

S2 When f can be approximated[9] with good probability, by a function g that has a low algebraic degree d.

More precisely, we assume that:

$$f(s_0, .., s_{n-1}) = g(s_0, .., s_{n-1}) \quad \text{holds:} \begin{cases} 1. \text{ with probability} \geq 1 - \varepsilon \\ 2. \text{ and with } g \text{ of degree } d. \end{cases}$$

[8] We describe an attack with a single boolean function f, still it is easy to extend it to stream ciphers using several different boolean functions.

[9] If such a (sufficiently good) approximation exists, there are efficient algorithms to find it. This problem is also known as "learning polynomials in the presence of noise", or as "decoding Reed-Muller codes". See for example [12, 9].

Note: In the first scenario S1, when f has just a low algebraic degree, it is known that the system can be easily broken given $\binom{n}{d}$ keystream bits. A successful example of this attack is described for example in [2]. In this paper we show that, since in S2, we do not need for the function to have a low algebraic degree (S1), successful attacks can be mounted given much less keystream bits, and with much smaller complexities. For example in Toyocrypt the degree of f is 63, but in our attacks, it will be approximated by a function of degree 2 or 4.

3.4 The Actual Attack

Given m bits of the keystream, we have the following m equations to solve:

$$\forall i = 1 \ldots m, \quad b_{t_i} = f\left(L^{t_i}(k_0, \ldots, k_{n-1})\right)$$

We recall that f, and all the L^{t_i} are public, and only the k_j are secret[10]. Each of the keystream bits gives one multivariate equation of degree d, with n variables $(k_0, .., k_{n-1})$ and being true with probability $(1-\varepsilon)$:

$$\forall i = 1 \ldots m, \quad b_{t_i} = g\left(L^{t_i}(k_0, \ldots, k_{n-1})\right) \quad \text{with probability} \quad \geq 1-\varepsilon$$

If we choose m such that $(1-\varepsilon)^m \geq \frac{1}{2}$, we may assume that all these equations are true and we have to find a solution to our system of m multivariate equations of degree d with n variables. More generally, even if $(1-\varepsilon)^m < \frac{1}{2}$, the attack still works, if we repeat it about $(1-\varepsilon)^{-m}$ times, each time for a different subset of m keystream bits, and until it succeeds. The complexity of this attack will be the complexity of generalized XL obtained in Section 2.2, multiplied by the number of repetitions necessary to succeed:

$$WF = T^{\omega}(1-\varepsilon)^{-m} \quad \approx \quad \binom{n}{n \,/ m^{1/d}}^{\omega} (1-\varepsilon)^{-m}$$

The above attack requires about m keystream bits, out of which we choose m at each iteration of the attack. We also need to choose m that minimizes the complexity given above. In practice, since the XL algorithm complexity increases increases by big leaps, with the value of D, we will in fact choose D and determine a minimal m for which the attack works.

4 Non-linear Filtering using Bent Functions

In order to prevent the numerous known fast correlation attacks, ciphers such as we described above (for example filter generators) should use a function f that is highly non-linear. For this, Meier and Staffelbach suggested at Eurocrypt'89 to use so called perfect non-linear functions, also

[10] Important: if L is not public, as it is may be the case in Toyocrypt, our later attacks will not work. Nevertheless they show that Toyocrypt is cryptographically weak.

known as "bent functions" [16, 22]. These functions achieve optimal resistance to the correlation attacks, because they have a minimum (possible) correlation to all affine functions, see Theorem 3.5. in [16]. It is therefore tempting to use a bent function as a combiner in a stream cipher. And indeed many cryptographic designs (e.g. Toyocrypt, and not only in stream ciphers) use such functions, or modified versions of such functions[11].
Unfortunately optimality against one attack does not guarantee the security against other attacks. Following Anderson [1], any criteria on f itself cannot be sufficient. The author of [1] claims that "attacking a filter generator using a bent or almost bent function would be easy" and shows why on small examples. He considers "an augmented function" that consists of α copies of the function f applied to consecutive windows of n consecutive bits, among the $n+\alpha$ consecutive bits of an LFSR output stream. He shows explicit examples in which even if $f : GF(2)^n \rightarrow GF(2)$ is a bent function, still the augmented function $GF(2)^{n+\alpha} \rightarrow GF(2)^\alpha$ will have very poor statistic properties, and thus will be cryptographically weak.

For real ciphers, it is difficult to see if Anderson's remark is really dangerous. For example in Toyocrypt, an MLFSR is used instead of an LFSR, which greatly decreases the number of common bits between two consecutive states, and more importantly, only a carefully selected subset of state bits is used in each application of f. Thus it seems that Toyocrypt makes any version of the attacks described by Anderson in [1] completely impractical.

Bent Function Used in Toyocrypt
The combining function f of Toyocrypt is built according to:

Theorem 4.0.1 (Rothaus 1976 [22]). Let g be any boolean function $g : GF(2)^k \rightarrow GF(2)$. All the functions $f : GF(2)^{2k} \rightarrow GF(2)$ of the following form are bent:

$$f(x_1, x_2, \ldots, x_{2k}) \quad = \quad x_1x_2+x_3x_4+\ldots,+x_{2k-1}x_{2k} + g(x_1, x_3, \ldots, x_{2k-1})$$

Remark: More precisely, the function of Toyocrypt is a XOR of s_{127} and a function built according to the above theorem. We must say that using such a function as a non-linear filter is **not** a very good idea. It is easy to see that if we use a single LFSR or MLFSR, there will be always a "guess and find" attack on such a cipher. This is due to the fact that if we guess and fix k state bits, here it will be the odd-numbered bits, then

[11] In general the authors of [16] did not advocate to use pure bent functions, because it is known that these functions are not balanced and cannot have a very high degree. They advise to use modified bent functions, for which it is still possible to guarantee a high non-linearity, see [16].

the expression of the output becomes linear in the other state bits. This can be used to recover the whole state of the cipher given $3k/2$ bits of it, i.e. the effective key length in such a scheme is only $3k/2$ instead of $2k$ bits. This attack is explained in details (on the example of Toyocrypt) in [18]. In this paper we do not use this property of f, and design a different attack, based on the low number of higher degree monomials, and thus being potentially able to break variants of Toyocrypt that are not based on the above theorem and for which there is no "guess and find" attacks.

5 Application of XL to the Cryptanalysis of Toyocrypt

In this section we present a general attack on Toyocrypt [18], a cipher that was, at the time of the design, believed to resist to all known attacks on stream ciphers. In Toyocrypt, we have one 128-bit LFSR, and thus $n = 128$. The boolean function is as follows:

$$f(s_0, .., s_{127}) = s_{127} + \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42} +$$

$$+ s_1 s_2 s_9 s_{12} s_{18} s_{20} s_{23} s_{25} s_{26} s_{28} s_{33} s_{38} s_{41} s_{42} s_{51} s_{53} s_{59} + \prod_{i=0}^{62} s_i.$$

with $\{\alpha_0, \ldots, \alpha_{62}\}$ being some permutation of the set $\{63, \ldots, 125\}$. This system is quite vulnerable to the XL higher order correlation attack we described above: there are only a few higher-order monomials, one of degree 4, one of degree 17 and one of degree 63. Everything else is quadratic.

A Quadratic Approximation

Most of the time, the system is quadratic. We put: $g(s_0, .., s_{127}) = \sum_{i=0}^{62} s_i s_{\alpha_i}$. Then $f(s) = g(s)$ holds with probability about $1-2^{-4}$. With the notations of the Section 3.4 we have $d = 2$ and $\varepsilon = 2^{-4}$. Currently, it is an open problem if this approximation allows any efficient attacks on Toyocrypt.

An Approximation of Degree $d = 4$

One can also see that if we put:

$$g(s_0, .., s_{127}) = \sum_{i=0}^{62} s_i s_{\alpha_i} + s_{10}s_{23}s_{32}s_{42}.$$

Then $f(s) = g(s)$ holds with probability very close to $1 - 2^{-17}$. We have $d = 4$ and we have approximatively $\varepsilon = 2^{-17}$.

5.1 Our Higher Order Correlation Attack on Toyocrypt

The equation $(1-\varepsilon)^m \approx \frac{1}{2}$ gives $m \approx 2^{16}$. This is simply to say that if we consider some 2^{16}, not necessarily consecutive bits of the keystream, the probability that for **all** of them we have $f(s) = g(s)$ will be about 1/2. A more precise evaluation shows that if we put $m = 1.3 \cdot 2^{16}$, we still have $(1-\varepsilon)^m = 0.52$. This is the value we are going to use.

Thus, given some m keystream bits, $m = 1.3 \cdot 2^{16}$, one can write from Toyocrypt m equations of degree 4 and with 128 variables k_i. To this system of equations we apply generalized XL as described in Section 2. We have $n = 128$ and let $D \in \mathbb{N}$. We multiply each of the m equations by all products of up to $D-4$ variables k_i. The number of generated equations is: $R = m\left(\sum_{i=0}^{D-4}\binom{n}{i}\right)$ We also have $T = \left(\sum_{i=0}^{D}\binom{n}{i}\right)$. We observe that for $D = 9$ we get $R/T = 1.1401$. Following our simulations and their analysis given in Section A.3, and since $D < 3d$, we expect that the exact number of linearly independent equations is $Free = \min(T, R - \binom{m}{2} - m) - \epsilon$ with a very small ϵ. This $Free$ is sufficient: we have $(R - \binom{m}{2} - m))/T = 1.13998$, and thus $R - \binom{m}{2} - m > T$ and $R - \binom{m}{2} - m$ is not very close to T. From this, following Conjecture A.3.1 and our simulation results, we expect that $Free = T - \epsilon$ with $\epsilon = 1$. XL works for D=9. The complexity of the attack is basically the complexity of solving a linear system $T \times T$ (we don't need to take more than T equations). With Strassen's algorithm, we get:

$$WF = \frac{7}{64} \cdot T^{log_2 7} = 2^{122}.$$

6 Improved XL Higher Correlation Attacks

We will now explore the tradeoff described in Section 3.4. The basic idea is that, if we diminish a little bit a success probability of the attack, we may use a higher m, the system will be more overdefined and we will be able to use a lower value of D. This in turn greatly diminishes the value of T that may compensate for the necessity to repeat the attack.

Improved Attacks Exploring the Tradeoff

In the attack above we saw that $Free = \min(T, R - \binom{m}{2} - m) - \epsilon$ and that we may in fact neglect $\binom{m}{2} - m$. Moreover if D becomes smaller, and when $D < 2d = 8$, following Section A.3 we expect to have $Free = \min(T, R) - 1$. Thus we may say that for $D < 9$, and $R > 1.1 \cdot T$ the attack does certainly work. It gives the following condition on m:

$$m\left(\sum_{i=0}^{D-4}\binom{n}{i}\right) > 1.1 \cdot \left(\sum_{i=0}^{D}\binom{n}{i}\right)$$

From this, given D, we put $m = 1.1\left(\sum_{i=0}^{D}\binom{n}{i}\right)/\left(\sum_{i=0}^{D-4}\binom{n}{i}\right)$. The probability that our approximation of degree 4 holds for all m equations is $(1-\frac{1}{2^{17}})^m$. Finally, the complexity of the whole attack is:

$$WF = (1-\frac{1}{2^{17}})^{-m}\cdot 7\cdot T^{log_2 7}/64 = (1-\frac{1}{2^{17}})^{-m}\cdot\frac{7}{64}\cdot\left(\sum_{i=0}^{D}\binom{n}{i}\right)^{log_2 7}$$

The number of keystream bits required in the attack is about m, and the memory is T^2 bits. In the following table we show possible tradeoffs:

D	4	5	6	7	8	9
Data	2^{23}	2^{21}	2^{19}	2^{18}	2^{17}	2^{16}
Memory	2^{89}	2^{56}	2^{65}	2^{73}	2^{81}	2^{88}
Complexity	2^{200}	2^{102}	$\mathbf{2^{96}}$	2^{102}	2^{112}	2^{122}

Now, our best attack is in 2^{96}, requires 2^{65} bits of memory and only 82 kilobytes of keystream.

Better Attacks with an Iterated Variant of XL

It is possible to improve this attack slightly by iterating the XL algorithm. Here is one possible way to do this. We start with $m = 1.6\cdot 2^{18}$ keystream bits. The probability that all the corresponding m approximations of degree 4 are true is $(1-\frac{1}{2^{17}})^m \approx 2^{-4.62}$. This means that the whole attack should be repeated on average $2^{4.62}$ times. Now we apply the XL algorithm with $D = 5$, i.e. we multiply each equation by nothing or one of the variables. We have $R = 129\cdot 1.6\cdot 2^{18}$. The goal is however not to eliminate most of the terms, but only all the terms that contain one variable k_0. Let T' be the number of terms in T that does not contain the first variable k_0. We have $T = \sum_{i=0}^{D}\binom{n}{i}$ and $T' = \sum_{i=0}^{D}\binom{n-1}{i}$. The number of remaining equations of degree $e = 5$ that contain only $n' = 127$ variables is $R-(T-T') = 129\cdot 1.6\cdot 2^{18} - \sum_{i=0}^{5}\binom{128}{i} + \sum_{i=0}^{5}\binom{127}{i} = 2^{25.37}$. We have $R'/(T-T') = 5.06$ and the elimination takes the time of $7\cdot T^{log_2 7}/64 = 2^{75.5}$. Then we re-apply XL for $d = 5$, $n' = 127$, $m' = R-(T-T') = 2^{25.37}$ and $D = 6$. We have $R'/T' = 1.021$ and XL works with the complexity of $2^{87.59}$.

The complexity of the whole attack is: $2^{4.62}\left(2^{75.5}+2^{87.6}\right) = 2^{92.2}$ CPU clocks. Our best attack is now in 2^{92}, it requires still 2^{65} bits of memory, and now only 51 kilobytes of keystream.

Comparison with Previously Known Attacks

Our new attack is much better than the generic purpose time/memory/data tradeoff attack described by Shamir and Biryukov in [23], that given the same number of keystream bits, about 2^{19}, will require about 2^{109} computations (in pre-computation phase).

Our attack is sometimes better, and sometimes worse than the Mihaljevic and Imai attack from [18]. In [18], given much more data, for example 2^{48} bits, and in particular at least some 32 consecutive bits of the keystream, and given the same quantity of memory 2^{64}, the key can be recovered with a pre-computation of 2^{80} and processing time 2^{32}.

However if the keystream available does not contain 32 consecutive bits, only our attack will work. Similarly, if the keystream available is limited to 2^{19} bits, both the Mihaljevic and Imai attack [18] and the generic tradeoff attack from [23] will require a pre-computation of about 2^{109}. In this case our attack in 2^{92} is better.

7 Extensions and Generalizations

Improved Elimination Methods. A careful implementation of our attack could be substantially faster. It is possible that there are more careful elimination algorithms, that generate the equations in a specific order and eliminate monomials progressively, so that they are not generated anymore. We also expect that fast Gröbner bases algorithms such as Faugère's F5/2 [8] would improve our attack, at least in practice.

Variants of Toyocrypt. Our XL-based attacks can cryptanalyse not only Toyocrypt but also many variants of Toyocrypt that resist to all known attacks. For example, if in Toyocrypt we replace the bilinear part of f by a random quadratic form, such "guess-and-find" attacks as in [18] are not possible anymore, still our XL-based higher degree correlation attack works all the same. The same is true when we leave the quadratic part unchanged and add to f some terms of degree 3 and 4 in variables $x_2, x_4, \ldots$. It is also possible to see that, if the positions of the known bits of the keystream are sparsely distributed, and we do not have any known 32 consecutive bits, the attacks from [18] will not work anymore, and our attack still works.

New Attack Scenarios S3 and S4. Since this paper was written, there was substantial progress in algebraic attacks on stream ciphers. Generalizing the attack scenarios S1 and S2 described in this paper, two new attack scenarios S3 and S4 have been introduced by Courtois and Meier [5]. The principle of these new attacks is (roughly) to generate new multivariate equations of substantially lower degree than the original ones, by multiplying the equations by well-chosen multivariate polynomials. Thus, the authors are able to break Toyocrypt in 2^{49} CPU clocks instead of 2^{92}, and also present an attack in 2^{57} for LILI-128.

8 Conclusion

In this paper we studied higher order correlation attacks on stream ciphers. Our approach is to reduce the problem of recovering the (initial) state of a cipher, to solving an overdefined system of multivariate equations. We studied the Toyocrypt stream cipher, accepted to the second phase of the Japanese government Cryptrec call for cryptographic primitives. It is a 128-bit stream cipher, and at the time of submission of Toyocrypt, it was claimed to resist to all known attacks on stream ciphers. The weakness of Toyocrypt we exploited here is the presence of only a few higher degree monomials. It has already been identified as dangerous in Rueppel's book [21], page 79, back in 1986, however the designers of Toyocrypt ignored this warning.

Having little higher degree monomials, it is possible to approximate the filtering function, by a function of a much lower degree with a good probability. From this we were able to reduce the cryptanalysis of Toyocrypt to solving a system of multivariate equations of degree 4. In order to solve it, we studied an extension of the XL algorithm proposed at Eurocrypt 2000 for the case of quadratic equations [24]. The problem about XL is that it is heuristic, not all equations that appear in XL are linearly independent, and thus it is somewhat difficult to say to what extent is works. In this paper we showed that we are always able to explain the origin of the linear dependencies that appear in XL and to predict the **exact** number of non-redundant equations in XL.

Our best higher order correlation attack on Toyocrypt requires 2^{92} CPU clocks for a 128-bit cipher. This is achieved using only 51 kilobytes of the keystream, that does not have to be consecutive, and using 2^{65} bits of memory. This attack will work in many scenarios in which all known attacks fail, for example when the plaintext in only partially known.

We conclude that higher order correlation immunity, should be taken more seriously than previously thought, in the design of stream ciphers.

Acknowledgements: This paper has been written following the initial idea suggested by David Wagner. I wish also to thank Willi Meier, Josef Pieprzyk and Greg Rose for helpful remarks, and I'm grateful to Mehdi-Laurent Akkar for writing some useful code for the simulations.

References

1. Ross Anderson: *Searching for the Optimum Correlation Attack*, FSE'94, LNCS 1008, Springer, pp 137-143.
2. Steve Babbage: *Cryptanalysis of LILI-128;* Nessie project internal report, available at `https://www.cosic.esat.kuleuven.ac.be/nessie/reports/`.

3. Paul Camion, Claude Carlet, Pascale Charpin and Nicolas Sendrier, *On Correlation-immune Functions;* In Crypto'91, LNCS 576, Springer, pp. 86-100.
4. Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions"; J. Symbolic Computation (1990), 9, pp. 251-280.
5. Nicolas Courtois and Willi Meier: *Algebraic Attacks on Stream Ciphers with Linear Feedback,* Eurocrypt 2003, Warsaw, Poland, LNCS, Springer.
6. Nicolas Courtois and Jacques Patarin, *About the XL Algorithm over* $GF(2)$*;* Cryptographers' Track RSA 2003, San Francisco, April 13-17 2003, LNCS, Springer.
7. Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations;* Asiacrypt 2002, LNCS 2501, Springer, a preprint with a different version of the attack is available at `http://eprint.iacr.org/2002/044/`.
8. Jean-Charles Faugère: *Computing Gröbner basis without reduction to 0,* Workshop on Applications of Commutative Algebra, Catania, Italy, April 2002, ACM Press.
9. Oded Goldreich, Ronitt Rubinfeld and Madhu Sudan: *Learning polynomials with queries: The highly noisy case,* preprint September 13, 1998. A preliminary version appeared in 36th Annual Symposium on Foundations of Computer Science, pages 294-303, Milwaukee, Wisconsin, 23-25 October 1995. IEEE.
10. Michael Garey, David Johnson: *Computers and Intractability, a guide to the theory of NP-completeness*, Freeman, p. 251.
11. Jovan Dj. Golic: *On the Security of Nonlinear Filter Generators,* FSE'96, LNCS 1039, Springer, pp. 173-188.
12. Jovan Dj. Golic: *Fast low order approximation of cryptographic functions,* Eurocrypt'96, LNCS 1070, Springer, pp. 268-282.
13. Mireille Martin-Deschamps, private communication.
14. James L. Massey, Rainer A. Rueppel: *Linear ciphers and random sequence generators with multiple clocks,* in Eurocrypt'84, LNCS 209, Springer.
15. Willi Meier and Othmar Staffelbach: *Fast correlation attacks on certain stream ciphers;* Journal of Cryptology, 1(3):159-176, 1989.
16. Willi Meier and Othmar Staffelbach: *Nonlinearity Criteria for Cryptographic Functions;* Eurocrypt'89, LNCS 4234, Springer, pp.549-562.
17. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: *Handbook of Applied Cryptography*; CRC Press.
18. M. Mihaljevic, H. Imai: *Cryptanalysis of Toyocrypt-HS1 stream cipher*, IEICE Transactions on Fundamentals, vol. E85-A, pp. 66-73, Jan. 2002. Available at `http://www.csl.sony.co.jp/ATL/papers/IEICEjan02.pdf`.
19. T.T. Moh: *On The Method of XL and Its Inefficiency Against TTM*, available at `http://eprint.iacr.org/2001/047/`.
20. Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms;* Eurocrypt'96, pp. 33-48.
21. Rainer A. Rueppel: *Analysis and Design of Stream Ciphers,* Springer, New York, 1986.
22. O. S. Rothaus: *On "bent" functions;* Journal of Combinatorial Theory, Ser. A, Vol. 20, pp. 300-305, 1976.
23. Adi Shamir, Alex Biryukov: *Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers;* Asiacrypt 2000, LNCS 2248, Springer, pp. 1-13.
24. 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.
25. Volker Strassen: *Gaussian Elimination is Not Optimal;* Numerische Mathematik, vol 13, pp 354-356, 1969.

A The Exact Behaviour of XL for $d \geq 2$.

Let $Free$ be the maximum number of equations that are linearly independent in XL algorithm. We will show how to compute $Free$ exactly and compare the results with computer simulations.

In all the simulations that follow, we pick a random system of linearly independent equations $y_i = f_i(x_0, \ldots, x_{n-1})$ of degree $\leq d$ (non-homogenous). Then we pick a random input $x = (x_0, \ldots, x_{n-1})$ and 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 $l_i(x_0, \ldots, x_{n-1}) = 0$, for $i = 1, \ldots m$.

A.1 The Behaviour of XL for $d = 2$ and $D = 3$.

By definition, $Free$ is smaller than R and cannot exceed T, see Section 2.1. Therefore:

$$Free \leq \min(T, R)$$

We have done various computer simulations with $d = 2$ and $D = 3$. In the following table we fix n and try XL on a random system of m linearly independent equations with growing m and with a fixed D.

d	2	2	2	2	2	2	2	2	2	2	2	2
n	10	10	10	10	10	20	20	20	20	20	64	64
m	10	14	16	17	18	20	40	50	60	65	512	1024
D	3	3	3	3	3	3	3	3	3	3	3	3
R	110	154	176	187	198	420	840	1050	1260	1365	33280	66560
T	176	176	176	176	176	1351	1351	1351	1351	1351	43745	43745
$Free$	110	154	174	175	175	420	840	1050	1260	1350	33280	43744

Figure 1: XL simulations for $d = 2$ and $D = 3$.

n number of variables.
m number of equations.
D we generate equations of total degree $\leq D$ in the x_i.
R number of equations generated (independent or not).
T number of monomials of degree $\leq D$
Free number of linearly independent equations among the R equations.
$\diamond$ XL will work when $Free \geq T - D$.

Results: For $d = 2$ and $D = 3$ we observe that most of the time[12] $Free = \min(T, R)$ and at any rate, we always have $Free = \min(T, R) - \epsilon$ with $\epsilon = 0, 1, 2$ or 3.

[12] $Free$ is bounded by two functions and most of the time it is just the minimum of their values. However around the point where the two graphics meet, we sometimes have a "smooth transition": we observe that $Free = \min(T, R) - \epsilon$ with $\epsilon = 0, 1, 2$ or 3. Here the smooth transition is visible for $d = 2$, $n = 10$, $m = 16$, $D = 3$.

A.2 The Behaviour of XL for $d = 2$ and $D = 4$.

When $D = 4$ we do not have $Free = \min(T, R)$ anymore. However most of the equations are still linearly independent.

d	2	2	2
n	10	10	10
m	5	10	11
D	4	4	4
R	280	560	616
T	386	386	386
$Free$	265	385	385

d	2	2	2	2	2	2
n	20	20	20	20	20	20
m	20	24	28	30	32	36
D	4	4	4	4	4	4
R	4220	5064	5908	6330	6752	7596
T	6196	6196	6196	6196	6196	6196
$Free$	4010	4764	5502	5865	6195	6195

d	2
n	40
m	128
D	4
R	105088
T	102091
$Free$	96832

Figure 2: XL with $d = 2$ and $D = 4$ (notations as on Fig. 1).

Results: From these simulations, it can be seen that $d = 2$ and $D = 4$ we have always:

$$Free = \min\left(T, R - \binom{m}{2} - m\right) - \epsilon \quad \text{with } \epsilon = 0, 1, 2 \text{ or } 3.$$

The fact that $Free = R - \binom{m}{2} - m - \epsilon$ when $R - \binom{m}{2} - m \leq T$, suggests that, in all cases, there are $\binom{m}{2} + m$ linear dependencies between the equations in R. We are able to explain the origin (and the exact number) of these linear dependencies. Let l_i be the equations taken formally (not expanded), and let $[l_i]$ denote the expanded expression of the left side of these equations as quadratic polynomials. Then we have:

$$l_i[l_j] = [l_i]l_j$$

For each $i \neq j$, the above equation defines a linear dependency between the equations of XL. This explains the $\binom{m}{2}$ dependencies.

Example: For example if $l_1 = x_1x_3 + x_4$ (which means that the equation l_1 is $x_1x_3 + x_4 = 0$) and $l_5 = x_2x_1 + x_4x_7$ then the notation $l_1[l_5] = [l_1]l_5$ denotes the following linear dependency between the $l_ix_jx_k$:

$$l_1x_2x_1 + l_1x_4x_7 = l_5x_1x_3 + l_5x_4.$$

There also other dependencies. They come from the fact that we have:

$$l_i[l_i] = l_i$$

This explains the remaining m dependencies. For example if $l_1 = x_1x_3 + x_4$ we obtain that: $l_1 = l_1x_1x_3 + l_1x_4$.

A.3 Tentative Conclusion on XL and More Simulations for $d \geq 2$ and $D \geq 4$.

From the above simulations, we see that, at least for simple cases, we are always able to predict the exact number of linearly independent equations that will be obtained. From the above simulations we conjecture that:

Conjecture A.3.1 (Behaviour of XL for $D < 3d$).

1. For $D = d \dots 2d-1$ there are no linear dependencies when $R \geq T$ and we have $Free = \min(T, R) - \epsilon$ with $\epsilon = 0, 1, 2$ or 3.
2. For $D = 2d \dots 3d-1$ there are linear dependencies and we have
$$Free = \min\left(T, R - \left(\sum_{i=0}^{D-2d} \binom{n}{i}\right)\left(\binom{m}{2} + m\right)\right) - \epsilon \text{ with } \epsilon = 0, 1, 2 \text{ or } 3.$$
The factor $(\binom{m}{2} + m)$ is due to the linear dependencies of type $l_i[l_j] = [l_i]l_j$ and $l_i[l_i] = l_i$ as explained above. Moreover when $D > 2d$ there are other linear dependencies that are products of these by monomials in x_i of degree up to $D - 2d$, and to count these we have multiplied their number by a factor $\left(\sum_{i=0}^{D-2d} \binom{n}{i}\right)$.
3. It is also possible to anticipate what happens for $D \geq 3d$. However, it is more complex, and in this paper we do not need to know this.

Theory vs. Practice Here is a series of simulations with different $d > 2$ and different values of D to see if our conjecture is verified in practice.

d	3	3	3	3	3	3
n	10	10	10	10	10	10
m	10	10	10	10	10	10
D	3	4	5	6	7	8
R	10	110	560	1760	3860	6380
T	176	386	638	848	968	1013
$Free$	10	110	560	846	966	1011

3	3	3	3	3
16	16	16	16	16
16	16	16	16	16
3	4	5	6	7
16	272	2192	11152	40272
697	2517	6885	14893	26333
16	272	2192	11016	26330

Figure 3: XL with $d = 3$ (notations as on Fig. 1).

d	4	4	4	4	4	4	4
n	10	10	10	10	10	10	10
m	10	10	10	10	10	10	10
D	4	5	6	7	8	9	10
R	10	110	560	1760	3860	6380	8480
T	386	638	848	968	1013	1023	1024
$Free$	10	110	560	966	1011	1021	1022

4	4	4	4	4
16	16	16	16	16
16	16	16	16	16
4	5	6	7	8
16	272	2192	11152	40272
2517	6885	14893	26333	39202
16	272	2192	11152	39200

Figure 4: XL with $d = 4$ (notations as on Fig. 1).

By inspection we see that these results, all our previous simulations, as well as those done in [6], always do confirm the Conjecture A.3.1.